Data General

Advanced Diagnostic EXecutive

ADEX

OPERATOR'S MANUAL

Data General Service, Inc.

A Subsidiary of Data General Corporation

ADVANCED DIAGNOSTIC EXECUTIVE

ADEX

Operator's Manual

014-000744-04

RECORD OF REVISIONS

DATE	REV.	CHANGE	NOTES/PAGES AFFECTED	APPROVAL
11/82	00	00	Original Issue	
03/83	01	00		
07/83	02	00		
11/83	03	00		
01/87	04	00	Changed the Advanced Diagnostic Executive System (ADES) to the Advanced Diagnostic EXecutive (ADEX); Updated and reformatted all sections.	

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

PREFACE

NOTE

The Advanced Diagnostic EXecutive ADEX was formally known as the Advanced Diagnostic Executive System (ADES).

This manual introduces Data General's Advanced Diagnostic EXecutive (ADEX). It explains how to:

- Load ADEX.
- Control the operating environment.
- Use ADEX to troubleshoot malfunctioning equipment.

The troubleshooting information is generic. For more specific details, refer to the appropriate maintenance manual.

This manual assumes you have no previous experience with ADEX, but you should be familiar with the following:

- The basic parts of your computer system.
- How to use an operating system to load and control other programs.
- How to operate switches on the CPU's front panel (or a soft console equivalent) of the computer system on which you are running ADEX.

The Structure of this Manual

This manual is divided into three chapters and seven appendices.

Section 1 – OPERATIONS – provides an overview of all the tasks you can perform with ADEX. This section tells you when to use particular commands and utilities, but does not describe them in detail. Sections 2 and 3 provide the details. Section 1 briefly explains:

- ADEX and its tests.
- Bringing up the Operating System.
- The System Equipment Table.
- Working with Media.
- Controlling the Operating Environment.
- The file System.
- Basic Troubleshooting.

Section 2 – ADEX–CLI – provides complete details on all Command Line Interpreter (CLI) commands. The first section gives a brief overview on using the CLI; and the second subsection describes all of the CLI commands in detail.

Section 3 – ADEX UTILITIES – provides complete details on the most frequently used ADEX Utilities. The following utilities are described in Section 3:

- ADEX Media Builder (AMB)
- Edit Utility (including the Edit commands)
- Octal Debugging Tool (ODT)
- Script Builder (SCRBLD)
- The Update Utility
- ADEX File Editor
- File Display Utility
- DTR Form Print Utility
- Contiguous Memory Test Utility (CMENT)
- Symbolic Debugger (ADEB)

The less frequently used ADEX utilities are described in Appendices F and G.

Appendix A –	Glossary
Appendix B –	Bootstrap Troubleshooting
Appendix C –	Panic Codes
Appendix D –	Mnemonic Descriptions
Appendix E –	CORESIDENT Installation Flowcharts
Appendix F –	Conditional CLI
Appendix G –	CLI Macros

Typesetting Conversions

When referring to commands, **UPPERCASE BOLD FACE** is used to distinguish them from other text.

Command lines appear as follows:

```
Type-CLI>          COMmand arg1   [arg2]
                                  [arg2[arg3]]
```

where:

Type-CLI>	is the prompt that informs you that a particular utility is running and awaiting input from you. Here, type may be ADEX or EDIT. For example, ADEX-CLI>.
COMmand	is the name of the command, with the uppercase letters indicating how you can abbreviate the command. For example, **DELete** means that when you use the **DELETE** command, you only have to type in **DEL** (although you can type the entire command, if you wish).
arg1	is an argument you must provide; for example, the name of a file. If you provide more than one argument, you must separate them with one or more spaces or one comma.
[ ]	are brackets indicating that the argument is optional. Never type in the brackets. They simply indicate that you have a choice. Note that you can have options within options. Also, you may have a choice of arguments. In this manual, choices are shown in vertical lists.
< >	are angle brackets indicating a list of arguments, separated by commas. You can choose one or more of these.

All numbers are in octal unless otherwise indicated.

NOTE

Depending on your system, one of two ADEX prompts will appear. For NE, MN, or ME ADEX, the prompt is ADEX-CLI>. For MV ADEX, the prompt is MV-ADEX>. ADEX-CLI> is used throughout this manual to refer to both prompts.

SECTION 1
OPERATIONS

SECTION 1
OPERATIONS

1.1 INTRODUCTION

ADEX, the Advanced Diagnostic EXecutive, lets you load and execute test programs. This diagnostic operating system is intended for use on all Data General processors. See the current release notice for a complete list of equipment and significant notes and patches that apply.

ADEX supports two System Exerciser test programs: SEARCH and MVSYSTEMX, as well as subsystem test programs.

ADEX also includes its own system programs and utility programs. The utilities allow you to perform many tasks, including the editing of program files, controlling the execution of tests, and updating or building more ADEX media.

1.1.1 SEARCH

SEARCH (System Exerciser and Reliability CHeck) is a diagnostic operating system that loads and runs programs to exercise CPUs and peripherals. SEARCH helps detect intermittent failures and interactive problems between two or more subsystems. It can also verify the repair and test the reliability of a hardware system.

The features of SEARCH include:

- Ability to build a SEARCH system disk.
- Can be run from either a disk or tape based SEARCH system.
- Automatically saves generated information on disk (when running disk-based system).
- Can save generated information on tape.
- SEARCH CLI Commands
- Resident ODT (Octal Debugger Tool)
- Testing communications equipment.
- Automatic saving of information at the time of an error for verification of either the fault or the repair (Disk only).
- Ability to reload SEARCH with either partial or no system generation.

- System–generation process that allows:
  a. SEARCH supervisor load area (either low or high end of the first 32K of memory).
  b. Automatic or manual sizing of computer system and peripherals.
  c. Manual sizing of communications equipment.
  d. List of tests to be run.
  e. Two execution modes for disk–based systems.

1.1.2 MVSYSTEMX

MVSYSTEMX is a system exerciser designed to run on DG MV series computers. The program is a stand–alone exerciser which exercises the system hardware architecture by approximating an operating system environment. MVSYSTEMX is a memory resident program that will concurrently exercise the host processor, memory subsystem, and attached peripherals.

1.1.3 ADEX Test Programs

The ADEX test programs include:

- Subsystem Diagnostic programs, which detect and isolate hard faults on subsystems and modules. Error reports indicate which Field Replaceable Unit (FRU) may be failing.
- Subsystem Reliability programs, which detect hard, intermittent, and interactive faults on subsystems and at lower levels. These programs are used to verify operation in a simulated user environment, and their error reports provide information about the environment in which the error is detected.
- Subsystem Verification programs, which quickly check that a product performs according to its functional specifications.
- Formatter programs, which initialize magnetic media.
- System Exerciser programs, which detect and isolate hard, intermittent, and interactive faults in systems and distributed systems. The error reports indicate which subsystem or FRU is failing.

1.1.4 System Programs

ADEX includes a number of system programs:

- Bootstrapping programs for loading ADEX into memory from either magnetic tape or disk.
- Driver programs that allow ADEX to communicate with peripherals.
- The Command Line Interpreter (CLI) which serves as an interface between you and the operating system.
- System utilities which are used by ADEX programs.
- Sizer programs for establishing the configuration of the system being testing.
- An edit utility for creating and editing files.

- A script builder for creating macros to test equipment.
- Debugger programs for tracking down software problems.
- A media builder for creating ADEX media on magnetic tape or disk.

1.1.5 General Procedures

Before ADEX can be used, it must be loaded into memory from tape or disk. This process is known as bootstrapping. The programs responsible for bootstrapping run some quick diagnostic tests, load in and inititalize the operating system, and transfer control to the Command Line Interpreter.

The CLI program is used to control the working environment, execute test programs, and run various utilities. Before starting any troubleshooting, some preparation is needed. For example, sizer programs must be run to set up the System Equipment Table (SET). This table contains the configuration of the system being tested. The CLI commands can be used to specify a different terminal as the master console and you can control testing from there. Finally, if working off tape media, it is recommended that an ADEX system is built on a disk for two reasons, to achieve response that it is substantially faster, as well as to be able to create and save files. If the system is already built, the system can be switched to or updated if necessary.

Once everything is set up, troubleshooting can be started. To test just the system configuration, use the RUN and ACCEPT macros. To test the communications subsystems, you have a choice of IAC, ISC, and TESTCOMM macros. Otherwise, a variety of commands allow the execution of test programs. Additionally, ADEX permits the use of macro files to execute programs. These files contain a sequence of commands that allow you to control the order in which the tests are run. A Script Builder utility lets you generate macro (script) files automatically, whereas an Edit utility gives more flexibility in the length and contents of macro files. The Edit utility also lets you patch any test programs that need updating and maintain a file of notes.

ADEX also includes more advanced troubleshooting facilities. For example, you can use a symbolic or octal debugger to see where test the program is failing. You can also control the actions of a program via a Switch Register (SWREG) or a String Buffer (STRING).

1.2 BRINGING UP ADEX

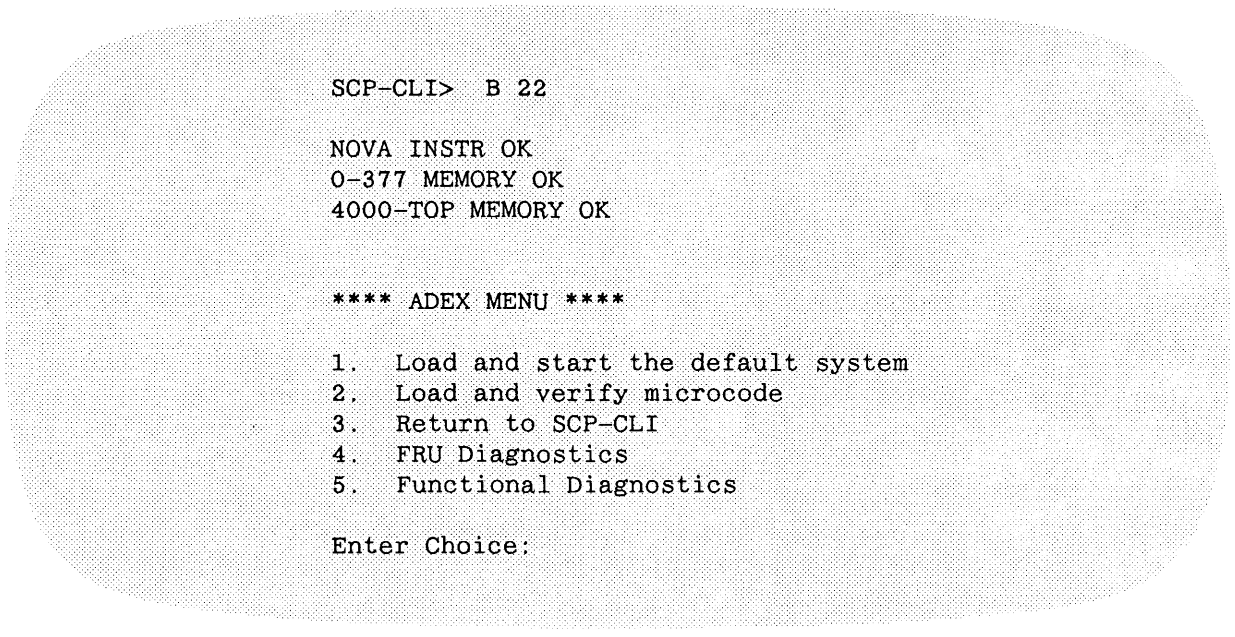
Bootstrapping is the initial process of loading ADEX from tape or disk into memory. The tape or disk containing ADEX is the ADEX system media. When loaded into memory and executed, it is known as the runtime media.

The program responsible for bootstrapping includes a test that checks the basic functions of the system hardware (excluding peripherals) during bootstrapping.

After bootstrapping, some initialization programs complete the process of loading the system. When successfully loaded, ADEX issues the ADEX> CLI prompt, indicating that you can start to enter CLI commands.

1.2.1 Bootstrapping

To bootstrap ADEX, follow the instructions in the appropriate documentation for the hardware system. This procedure varies from system to system. If bootstrapping is successful and the system hardware passes the test given by the ADEX bootstrap program, the message in Figure 1-1 will appear. Figure 1-1 View A is the message that appears for MVADEX. Figure 1-1 View B shows a NOVA®/ECLIPSE®, microECLIPSE™, or microNOVA® bootstrapping message.

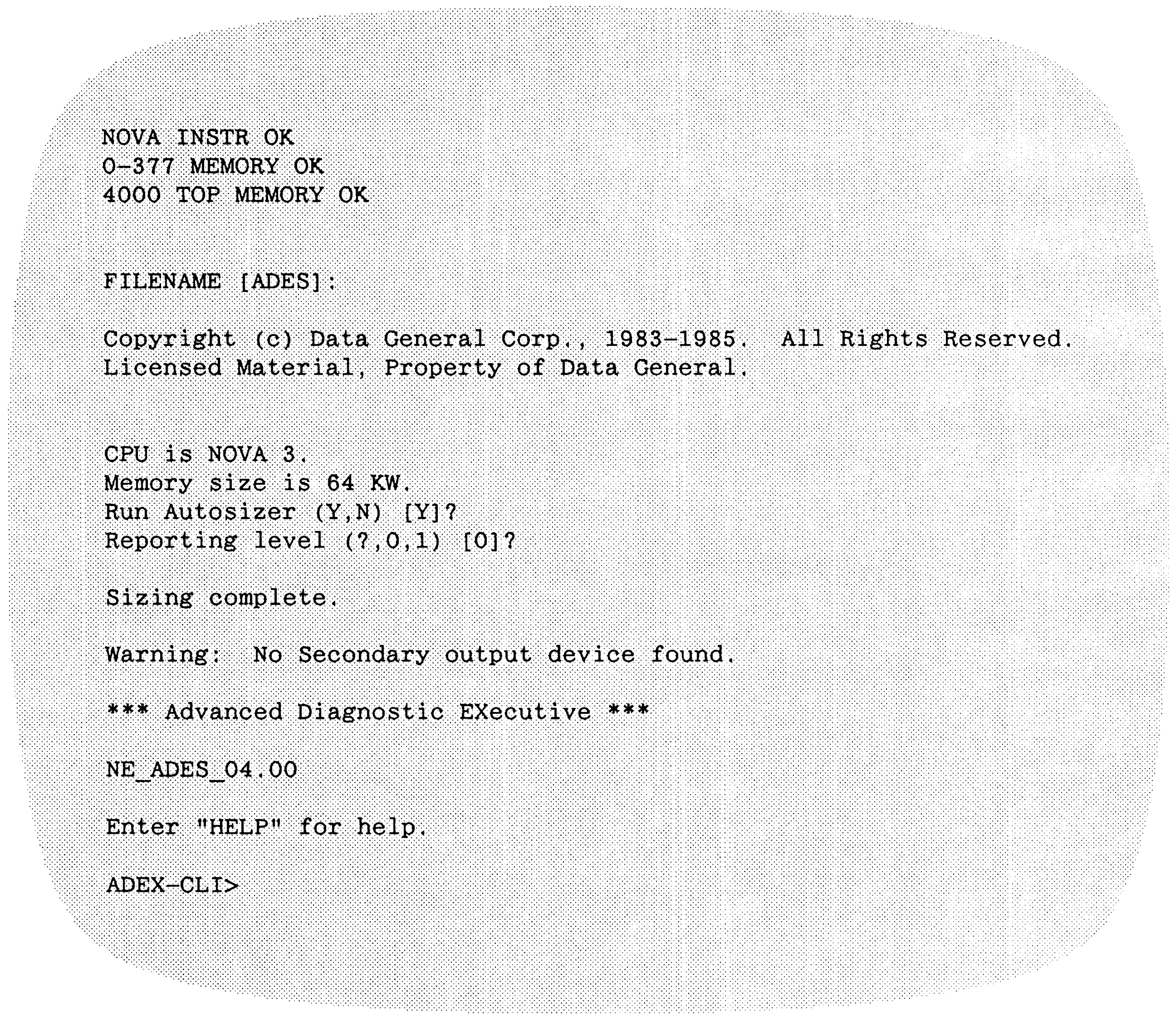


```
SCP-CLI>  B 22

NOVA INSTR OK
0-377 MEMORY OK
4000-TOP MEMORY OK


**** ADEX MENU ****

1.  Load and start the default system
2.  Load and verify microcode
3.  Return to SCP-CLI
4.  FRU Diagnostics
5.  Functional Diagnostics

Enter Choice:
```

FS-12826

Figure 1-1. (View A) Bootstrapping Message (MV ADEX)

- Option 1– Load and start the default system – Allows any device to be booted. Default is to the system disk.
- Option 2 – Load and verify microcode – This is for new systems only (MV/2000, MV/20000, MV/7800 etc.).
- Option 4 – FRU Diagnostics – This is for MV/2000, MV/7800, MV15000, and MV/20000 systems only.



```
NOVA INSTR OK
0-377 MEMORY OK
4000 TOP MEMORY OK


FILENAME [ADES]:

Copyright (c) Data General Corp., 1983-1985.  All Rights Reserved.
Licensed Material, Property of Data General.


CPU is NOVA 3.
Memory size is 64 KW.
Run Autosizer (Y,N) [Y]?
Reporting level (?,0,1) [0]?

Sizing complete.

Warning:  No Secondary output device found.

*** Advanced Diagnostic EXecutive ***

NE_ADES_04.00

Enter "HELP" for help.

ADEX-CLI>
```

FS-12827

Figure 1-1. (View B) Bootstrapping Message (NOVA/ECLIPSE, microECLIPSE, or microNOVA ADEX)

NOTE

Figure 1-1 View B contains the Bootstrapping, Loading and Initializing, and Auto Sizing messages.

If the test fails, the CPU is halted before the words are completely displayed. If the message does not appear within thirty seconds, some problem had been detected. See Appendix B for more detailed information.

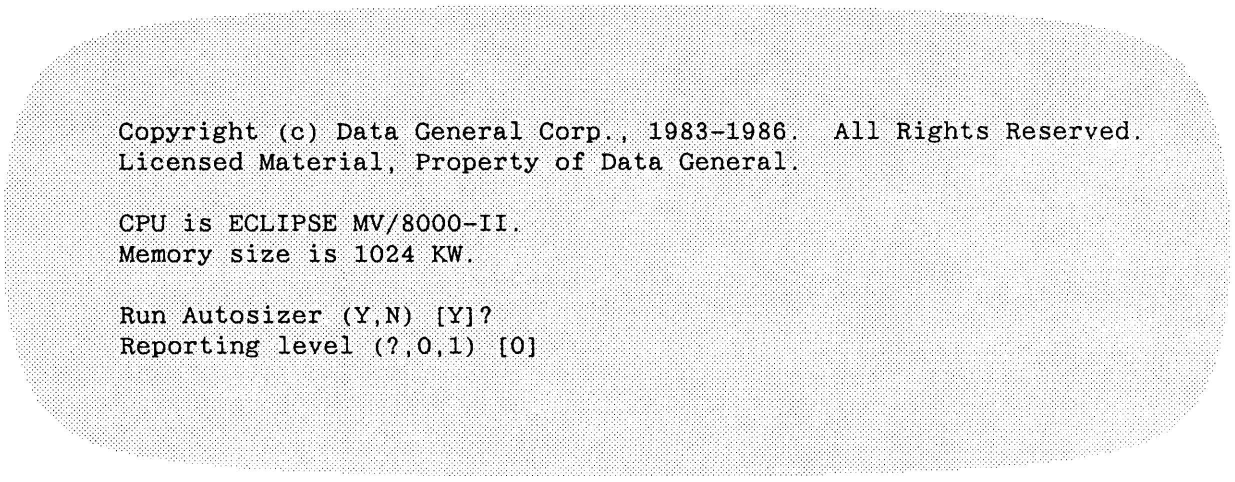
1.2.2 Initializing ADEX

If the full Bootstrapping message appears, you can load and execute either ADEX or a specialized stand-alone program. Generally, the stand-alone programs are used when the system lacks sufficient memory. ADEX requires a minimum of 16 (32) Kilowords (KW) for all 16-bit systems and 128 KW for 32-bit (MV) systems.

To load a stand-alone program, type in its name, then press NEW LINE. To load ADEX, select option 4 or 5 for MV/ADEX or NEW LINE for all other versions. (Remember that the square brackets around the word ADEX indicate that ADEX is the default answer to the question.)

When ADEX is loaded, a program called the Initialization Monitor takes over. This program initializes some system variables. From this point, a fatal ADEX error message may appear (also known as a system panic), should the operating system encounter a problem. For details on system panics, see Appendix C.

If the initialization Monitor encounters no problems, the System Initialization program takes over and concludes the initialization sequence. A message about the copyright, CPU, and the size of memory, as shown at the top of Figure 1-2 appears.



```
Copyright (c) Data General Corp., 1983-1986.  All Rights Reserved.
Licensed Material, Property of Data General.

CPU is ECLIPSE MV/8000-II.
Memory size is 1024 KW.

Run Autosizer (Y,N) [Y]?
Reporting level (?,0,1) [0]
```

FS-12828

Figure 1-2. Loading and Initializing Sequence

1.2.3 Sizing Peripherals

The next question asks whether the user wants to run the Auto Sizer program. This program sizes (i.e., identifies) all the peripherals it can find and adds the information to the System Equipment Table, which contains the configuration of the system.

Run the Auto Sizer if:

- You are loading ADEX from tape or have not previously saved a disk file called EQUIP0.
- You want to check quickly whether all the peripherals are working. If the Auto Sizer cannot find a peripheral, that peripheral may either be malfunctioning or off-line.

If running the Auto Sizer, then you must select the appropriate reporting level (0 or 1). See Figure 1–2. If you chose a reporting level of 0, two messages will be printed out. One message tells you that sizing is being done. The other messages states that it is completed. If you indicate a reporting level 1, then the Auto Sizer prints out a list of device codes and mnemonics for the devices the Auto Sizer is looking for. If the program finds a device, it assigns an F after the code. For example, MTA 22F 62 means that the Auto Sizer found a tape drive on device code 22, but not one on 62. See Figure 1–2. (Typing in a question mark means that the listed choices will be explained.)

NOTE

For tape and disk drives, an F after the device code means that at least a controller has been found. There may or may not be on–line units attached to the controller.

The default reporting level is 0. If you type in a question mark followed by NEW LINE or and illegal reponse, the following explanation is printed:

```
0 = SUPRESS AUTOSIZE OUTPUT.
1 = PRINT MNEMONICS AND DEVICE CODES AS THEY ARE SIZED.
```

For more information on the Auto Sizer, see the Auto Sizer subsection.

If, however, you are loading ADEX from disk and have previously built a System Equipment Table and saved it in a file called EQUIP0, it is not necessary to run the Auto Sizer. In this case, the System Initializer looks for EQUIP0 to use as the System Equipment Table (SET). If the program finds EQUIP0, a message says so. Otherwise, there is a warning that says the equipment table is empty. If warning is not received, it builds a System Equipment Table, as explained in the System Equipment Table subsection.

Once the system equipment table is set up, the System Initializer program looks for a description of the system console in the SET. If it does not find an entry in the SET describing the system console, the program assumes that it is a hardcopy device. To change the system console to a video terminal, follow the procedure for updating the System Equipment Table, after getting the CLI prompt.

The System Initializer now tries to find the secondary output device in the SET. If the program finds it, it sends a message, as in Figure 1–3. Otherwise, it sends a warning message. If this happens and a printer will be used as a secondary output device, make sure it is on–line. Then, after receiving the CLI prompt, run the Auto Sizer so that ADEX knows of the printer's existance. Finally, use the **SECONDARY** and **ENVIRONMENT** commands.

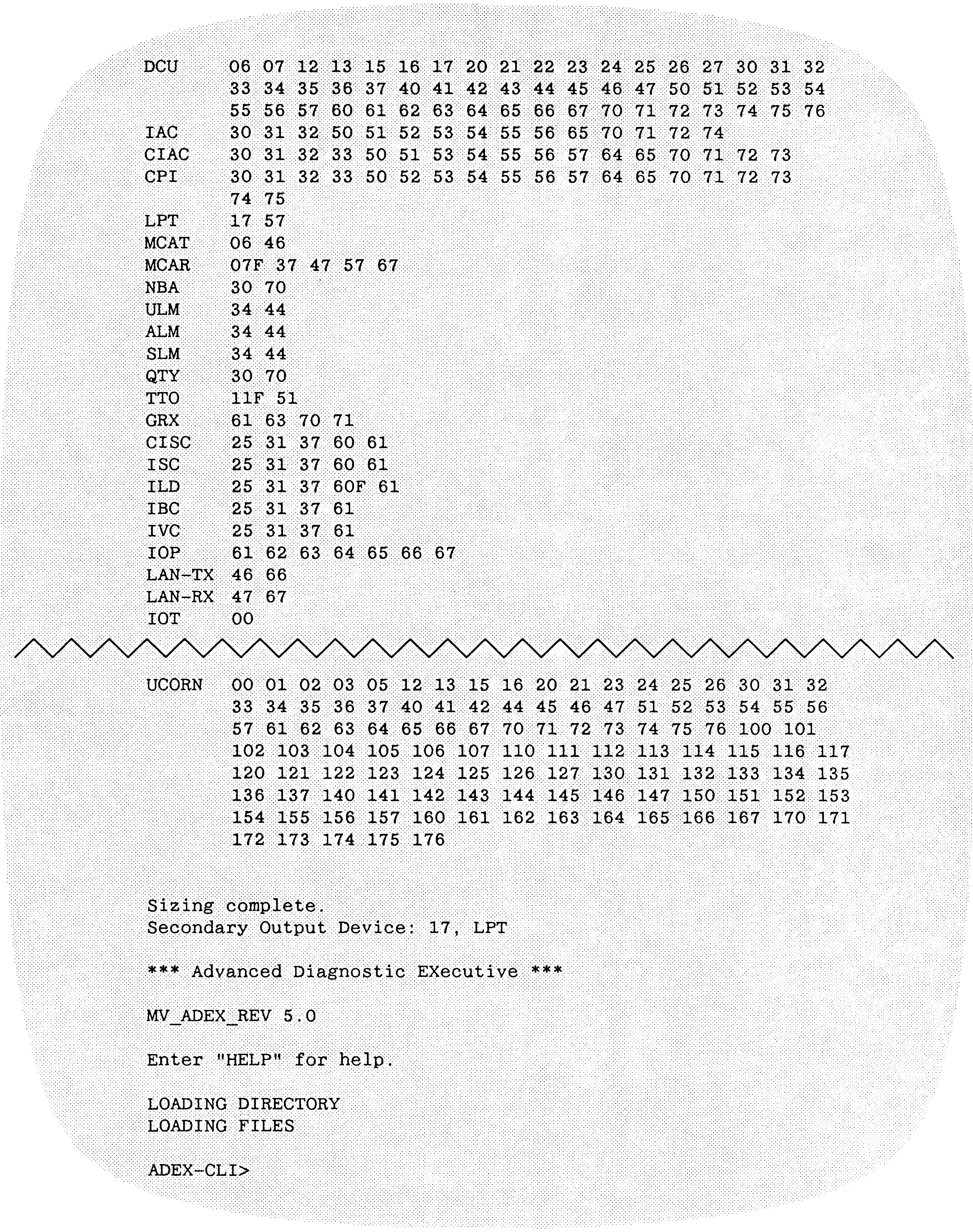
As a final step, the RISE macro is executed if it exists on the runtime media. This macro exists only if it has been previously created. RISE is defined as any other macro would be defined. It is particularly useful if the user knows just what sequence of commands needs to be executed when ADEX is first brought up. The RISE macro ensures that this sequence is carried out automatically.

Then the CLI prompt appears:

```
ADEX–CLI>
```

NOTE

The Auto Sizer and Manual Sizer are discussed in detail in subsections 1.3.1 and 1.3.2.



```
DCU     06 07 12 13 15 16 17 20 21 22 23 24 25 26 27 30 31 32
        33 34 35 36 37 40 41 42 43 44 45 46 47 50 51 52 53 54
        55 56 57 60 61 62 63 64 65 66 67 70 71 72 73 74 75 76
IAC     30 31 32 50 51 52 53 54 55 56 65 70 71 72 74
CIAC    30 31 32 33 50 51 53 54 55 56 57 64 65 70 71 72 73
CPI     30 31 32 33 50 52 53 54 55 56 57 64 65 70 71 72 73
        74 75
LPT     17 57
MCAT    06 46
MCAR    07F 37 47 57 67
NBA     30 70
ULM     34 44
ALM     34 44
SLM     34 44
QTY     30 70
TTO     11F 51
GRX     61 63 70 71
CISC    25 31 37 60 61
ISC     25 31 37 60 61
ILD     25 31 37 60F 61
IBC     25 31 37 61
IVC     25 31 37 61
IOP     61 62 63 64 65 66 67
LAN-TX  46 66
LAN-RX  47 67
IOT     00

UCORN   00 01 02 03 05 12 13 15 16 20 21 23 24 25 26 30 31 32
        33 34 35 36 37 40 41 42 44 45 46 47 51 52 53 54 55 56
        57 61 62 63 64 65 66 67 70 71 72 73 74 75 76 100 101
        102 103 104 105 106 107 110 111 112 113 114 115 116 117
        120 121 122 123 124 125 126 127 130 131 132 133 134 135
        136 137 140 141 142 143 144 145 146 147 150 151 152 153
        154 155 156 157 160 161 162 163 164 165 166 167 170 171
        172 173 174 175 176

Sizing complete.
Secondary Output Device: 17, LPT

*** Advanced Diagnostic EXecutive ***

MV_ADEX_REV 5.0

Enter "HELP" for help.

LOADING DIRECTORY
LOADING FILES

ADEX-CLI>
```

FS-12829

Figure 1-3. Sample Auto Sizer Output (Incomplete)

1.2.4 ADEX Command Line Interpreter

The ADEX Command Line Interpreter (CLI) lets you interact with the operating system. The CLI lets you control a number of system variables, such as the system console, the radix used by the operating system, and the way errors are reported by test programs. The CLI also lets you manipulate the many different kinds of files that ADEX supports. In addition, CLI commands let you execute system utilities (such as the Edit utility or Script Builder), as well as test programs.

The CLI can be used in two modes, manual and script. In manual mode the CLI processes commands that are entered in response to the CLI prompt, ADEX-CLI>. In script mode, the CLI processes commands that it finds in a script or macros file that was previously created. Macros can save a lot of time when frequently executing a sequence of commands. Instead of typing in the sequence each time, type only the name of the macro file in response to the CLI prompt. The CLI then executes all the commands in the file.

To enter a CLI command, type in the command name. Some commands require one or more pieces of information, known as arguments. If providing an argument, leave a space or type a comma between the command and the argument. Finally, to transmit the command, press the NEW LNE, Carriage Return, or LINE FEED key.

1.2.5 HELP Commands

ADEX-CLI includes a **HELP** command, plus a tutorial mode if help is needed in using the operating system. The **HELP** command provides information about CLI commands, related topics, system programs, and test programs. **HELP** can be used whenever manual mode under ADEX CLI is being used. To see a list of items that can be found in the HELP files, just type:

ADEX-CLI> **HELP**

NOTE

HELP files are normally only available on tape runtime media.

You can get information on all CLI commands, programs residing on the runtime media, and the following topics. The topics in the HELP facility include:

CCHARS	Control characters and keystroke commands
CFLIST	ADEX media builder custom file lists
CLI	All ADEX CLI commands
DEVICE_STATUS	Device status words
FTYPE	ADEX file types
MACROS	ADEX standard macros
MNEMONICS	ADEX mnemonics and model numbers
PANICS	ADEX panic codes
SEDIT	Screen edit utility
SYNTAX	Syntax conventions used in help files
UTILITIES	List of available ADEX utilities

To obtain information about a specific item type:

ADEX-CLI> **HELP** *item*

where *item* is the topic of the information needed. For example, the item may be the name of a program, a command, or a topic.

ADEX CLI also provides tutorial help whenever a system flag called OPERATOR is on and arguments to a command which requires them is omitted. The CLI then steps through a tutorial session so that you understand the arguments that must be provided. When bringing up ADEX, OPERATOR is automatically on so that tutorial help is automatically available unless you turn off this flag.

1.2.6 Failure to Load ADEX

If ADEX fails to load the first time, any number of system failures may be the cause. First, complete this checklist to be sure that:

- The CPU power is on.
- The CPU is unlocked.
- The system operator's console is on-line.
- The system media is on-line and at BOT (if tape) or recalibrated (if disk).
- All the cables are in good condition and plugged in properly.

If the answer to any of these is no, correct the situation and try loading ADEX again.

If none or only part of the bootstrap message appears, see Appendix B for more information. If a complete Test Message appears, but you encounter a system panic during initialization, go to Appendix C for more information. If, in either case, the hardware is not suspected, try a different media. For more information, refer to the appropriate maintenance manual for your system.

1.3 THE SYSTEM EQUIPMENT TABLE

The System Equipment Table (SET) is a table in memory that lists all the peripherals and options in the system. The SET includes information about the model numbers, device codes, and unit numbers of peripherals, as well as the CPU type and memory size. The table also lists the type of media and its write protection status. For disks, the dimensions are included. This kind of information is crucial to the diagnostic, reliability, and exerciser programs for peripherals.

To set up or modify the System Equipment Table, use the Auto Sizer (ASIZE) program, the Manual Sizer (MSIZE), or both. The Auto Sizer cannot size all peripherals (notably communications ones) and may be unable to correctly size malfunctioning peripherals. Also, the Auto Sizer program makes a number of assumptions about device-dependent information. Such information could be line characteristics information on an Intellegent Asynchronous Controller (IAC). So, generally, both sizer programs should be run to set up the System Equipment Table completely.

1.3.1 Auto Sizer (ASIZE)

You can run the Auto Sizer either when bringing up ADEX or any time after you have a CLI prompt. The Auto Sizer (ASIZE) automatically sizes all the peripherals that it finds and adds the information to the System Equipment Table.

To execute the program, type:

```
ADEX-CLI> XEQ ASIZE
```

or,

```
ADEX-CLI> ASIZE
```

ASIZE produces a list of device codes and mnemonics. For any device that it finds, the Auto Sizer adds an F after the device code. Figure 1-3 shows a sample list. Examine this list carefully to see if the program is omitting any devices. The **EQUIPMENT** command can also be used to check the contents of the SET. If any peripherals are missing, use the Manual Sizer to enter information about the peripherals.

NOTES

The Auto Sizer does not save the information in the SET on the disk. To do so, use the **EQUIPMENT WRITE** command.

For tape and disk drives, an F after the device code means that at least a controller has been found. There may or may not be on-line units attached to the controller. See Figure 1-3.

1.3.2 Manual Sizer (MSIZE)

The Manual Sizer (MSIZE) lets you manually add information to the SET about any peripherals in the system. Generally, it is best to use MSIZE to add any peripherals that the ASIZE has omitted. For example, if the System Initializer did not find a secondary output device, update the System Equipment Table using MSIZE. Similarly, of ASIZE assumed that the system console was a hardcopy terminal, it is possible to redefine it as a video console by using MSIZE.

To run the Manual Sizer type:

```
ADEX-CLI> MSIZE
```

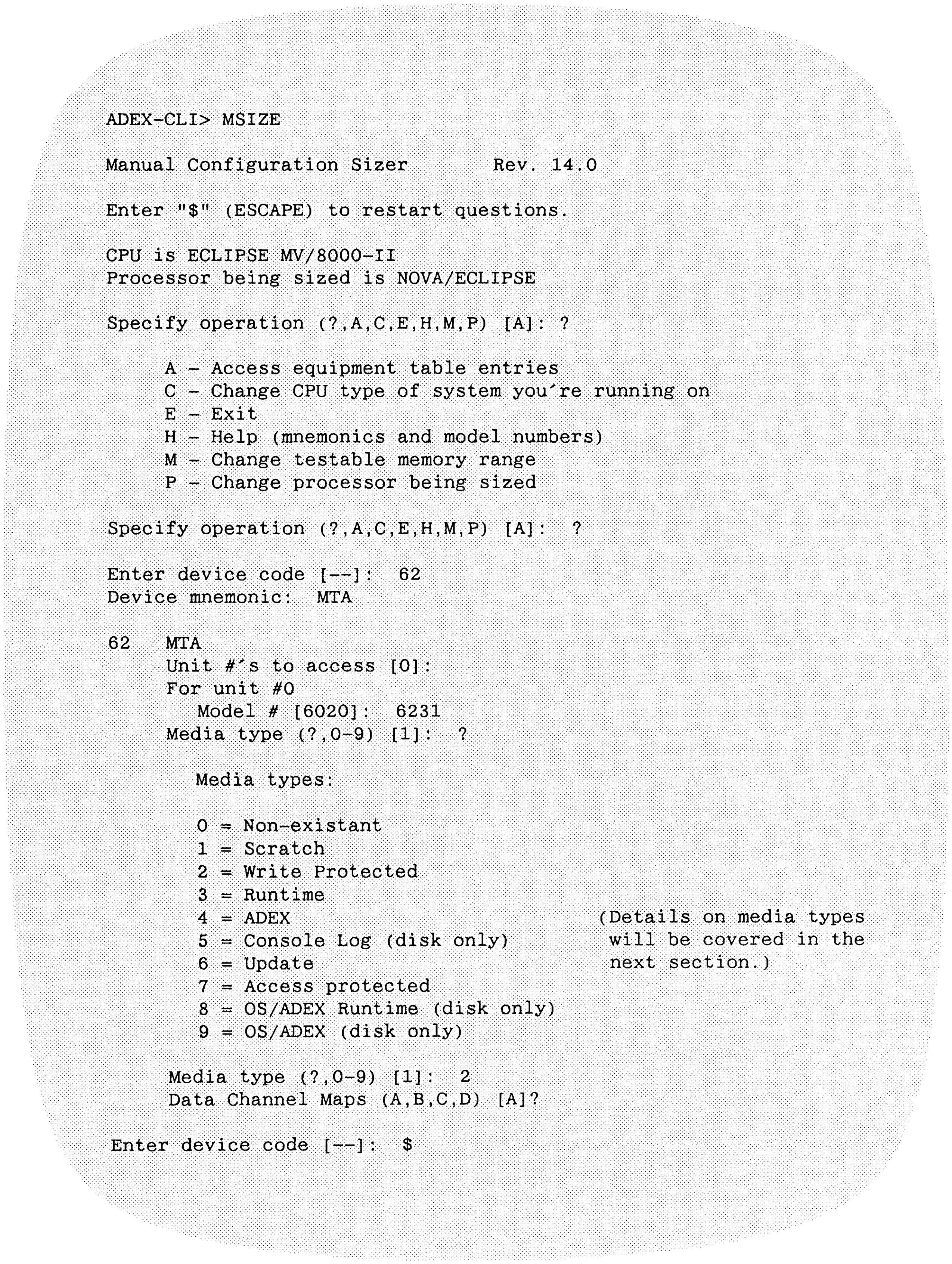
NOTE

If the operator flag is off, the program issues an error and returns to ADEX CLI. Use the **OPERATOR** command to turn OPERATOR on.

The manual sizer is self-documenting, so just follow the instructions given by the program. When you execute the program, you are asked a series of questions. These questions must be answered so that the program can enter the correct information. See Figure 1-4 for a sample output from MSIZE. To abort from MSIZE and return to ADEX-CLI, type CTRL-C, CTRL-A.

NOTE

1. If you hit ESC in answer to a question about device-dependent information, and you created the entry, then the entry is deleted and you get the ENTER DEVICE CODE question again.
2. You may only specify peripherals that run on the bus (processor) currently being sized (e,g,, you cannot specify a 6095 (MicroProducts Disk Drive) if the processor being sized is a NOVA® or ECLIPSE®).



```
ADEX-CLI> MSIZE

Manual Configuration Sizer       Rev. 14.0

Enter "$" (ESCAPE) to restart questions.

CPU is ECLIPSE MV/8000-II
Processor being sized is NOVA/ECLIPSE

Specify operation (?,A,C,E,H,M,P) [A]: ?

     A - Access equipment table entries
     C - Change CPU type of system you're running on
     E - Exit
     H - Help (mnemonics and model numbers)
     M - Change testable memory range
     P - Change processor being sized

Specify operation (?,A,C,E,H,M,P) [A]:  ?

Enter device code [--]:  62
Device mnemonic:  MTA

62   MTA
     Unit #'s to access [0]:
     For unit #0
        Model # [6020]:  6231
     Media type (?,0-9) [1]:  ?

        Media types:

        0 = Non-existant
        1 = Scratch
        2 = Write Protected
        3 = Runtime
        4 = ADEX                          (Details on media types
        5 = Console Log (disk only)        will be covered in the
        6 = Update                         next section.)
        7 = Access protected
        8 = OS/ADEX Runtime (disk only)
        9 = OS/ADEX (disk only)

     Media type (?,0-9) [1]:  2
     Data Channel Maps (A,B,C,D) [A]?

Enter device code [--]:  $
```

FS-12830

Figure 1-4. Sample MSIZE Output

1.3.3 Saving, Updating, and Clearing the SET

Since the SET is a table in memory, when you bring down ADEX the SET is gone, if you are running from tape media. Therefore, once you have run the sizer programs, consider using the CLI **EQUIPMENT** command, which allows you to save the SET information in a disk file called EQUIP0. This file saves you from having to run the sizer programs each time you bring up ADEX. If you have saved the information from SET in EQUIP0, you can avoid running the Auto Sizer altogether and have the System Initializer set up the SET with the information from EQUIP0.

In addition to letting you save the SET, the **EQUIPMENT** command serves other useful functions. It allows you to:

- Display the contents of the System Equipment Table.
- Save the SET in a file of your choice (a file other than EQUIP0).
- Replace the contents of the SET with either the contents of EQUIP0 or a file of your choice which contains a SET.
- Clear the contents of the SET (but preserve any versions saved on disk).

Three other CLI commands are useful in working with the System Equipment Table:

- **PROTECT**
- **UNPROTECT**
- **SCRATCH**

PROTECT and **UNPROTECT** set up and eliminate software write-protection, respectively, for a disk or tape. This information is stored in the SET. **SCRATCH** is used to change the media type of any disk or tape to Scratch Media.

1.4 MEDIA

The runtime media is the media from which ADEX is currently executing. A system media or ADEX media is any media (disk or tape) which contains ADEX software. However, at any point a system can become the runtime media.

You can find out what the runtime media is with either the **MEDIA** or **STATUS** commands. You can also use the **MEDIA** command to make a system media the runtime media.

Nine types of media exist:

- Scratch
- Write-protected
- Runtime
- ADEX
- Console Log (Tape Only)
- Update
- Access Protected
- OS/ADEX Runtime (Disk only)
- OS/ADEX (Disk only)

A scratch media is any device media available for testing. The data on scratch media is vulnerable. In contrast, as its name suggests, write-protected media cannot be written to by any .PRG type ADEX program. Generally, any customer disk packs with DGC software are write-protected. When ADEX comes up, unknown media are classified as write-protected (unless bit 1 in the System Status Word is 1, in which case unknown media are classified as scratch. A console log media is a tape which holds the console log. Finally, the update media contains the update files.

WARNING

Software write-protection restrictions do not apply to DTOS programs. To protect magnetic media, make sure the device is off-line before running DTOS programs.

NOTE

DTOS (Diagnostic Troubleshooting Operating System) was previous diagnostic operating system, now superceded by ADEX.

ASIZE automatically recognizes a number of magnetic media types. Table 1-1 lists those types (with numbers) and an explanation of their meaning. Figure 1-4 shows an example of how MSIZE lists media types by item number.

Table 1-1. Media Types

ITEM	TYPE	DESCRIPTION
0	Non-existant	A manual option only, used at the discretion of the local service representative.
1	Scratch	Write enabled test media.
2	Write protected	ADEX cannot write to (unknown) media.
3	Runtime	Tape or disk media currently executing.
4	ADEX	ADEX media, currently not executing.
5	Console Log	Tape (scratch) or disk runtime media for logging.
6	Update	Special 96-TPI diskette update files or AOS update format tapes.
7	Access protected	OS/ADEX Runtime mode only, used to disable remote Read/Write access to disks.
8	OS/ADEX Runtime (Disk only)	CORESIDENT media currently executing.
9	OS/ADEX (Disk only)	CORESIDENT media currently not executing.

At times, you may want to build a new system media. Running ADEX off tape has its disadvantages. It can be slower than disk. You lose the ability to create and save files. And, you cannot run a full set of test programs in an automatic test sequence. You can eliminate these restrictions by building and then running from disk. If you need to distribute a system easily, however, you may want to build it on tape. The ADEX Media Builder (AMB) lets you do both.

1.4.1 Building Media

There are two runtime media modes, stand-alone and CORESIDENT. ADEX resides on a separate disk in the stand-alone mode. It resides on the same disk as the operating system in CORESIDENT mode. The ADEX Media Builder can create a system on scratch media (a media that is not write-protected), on an ADEX media, and on a CORESIDENT system disk (a disk which contains the customer's software and has a designated ADEX area). You can check on the status of the media with the **EQUIPMENT** command. On the standalone media, if the media is write-protected, you can change its status by using the **UNPROTECT** or **SCRATCH** commands, or by running the Manual Sizer (MSIZE) and changing the status of the device to scratch, ADEX, or OS/ADEX as appropriate.

The ADEX Media Builder (AMB) operates in two modes: Automatic and Manual. Automatic mode is useful if you do not require operator input and want to include all the files which make up the runtime media in the media you are building. In automatic mode, the Media Builder builds a system on the first scratch media or OS/ADEX it finds in the System Equipment Table that can build a media of the model number specified. (The table is ordered by device code and unit number). The Builder copies all the files from the runtime media to the build media. If you want to control which device the Media Builder uses, you must make sure that all the devices ahead of it in the SET have write protection.

In contrast, manual mode is used if you want to select only a subset of the enabled files for the new media, if you want to build to a scratch device other than the first applicable scratch device listed in the SET, or if you want to build more than one media at a time.

In either mode, you can customize the new media. The files transferred to the build media can be customized in two ways:

- By selecting files and/or programs based on the contents of a System Equipment Table.
- By selecting files contained in a specified TXT file.

1.4.2 Updating Media

To allow you to update an ADEX system disk without rebuilding it, ADEX includes an Update Feature. Special update media is no longer distributed by Data General, as this feature now uses the standard ADEX released media to update disk runtime media.

New revisions are distributed by Data General Corporation to qualified CMOs, but users also have the option of creating their own special update media with AOS commands. This is especially beneficial for users who write their own diagnostics and wish to include their own files in the update.

NOTE

Separate Update Media is only sent out for MV/ADEX diskettes (30200-G) and MV/2000 ADEX (31291-G) and is only released with normal revisions.

1.5 CONTROLLING THE OPERATING ENVIRONMENT

Before you start running any programs or utilities, you should make sure your operating environment is set up the way you want it. A number of CLI commands let you display or set system variables, such as:

- Output devices (system console and secondary devices)
- System identification
- System defaults
- Error class for CLI commands
- Radix
- Environment Control Word (controls many aspects of the operating environment.

1.5.1 Output Devices

If you wish to switch the system console to another device, use the **CONSOLE** command. To switch and enable the secondary device, use the **SECONDARY** and **ENVIRONMENT** commands (change the SECONDARY flag), respectively. You can also use the **CONSOLE** and **SECONDARY** commands to find out what the system console and secondary output devices are.

1.5.2 System Identification

When using disk media, the CLI gives you the option of establishing an 80-character system identification message. This is very useful for identifying the runtime media. For example, your message could include information about the date and time when the media was built or the model number of the system. To set up or display a message, use the **SYSID** command.

NOTE

You cannot change the system identification when the runtime media is a tape.

1.5.3 System Defaults

You can set certain defaults with the **DEFAULT** command. For example, you can determine how many passes a test program makes, if the pass count is not specified on the **XEQ** command line (the initial setting is 1).

1.5.4 Error Class

Initially, the error class for CLI commands is set to ERROR. This means that if the CLI encounters a problem, it does not execute the rest of a command line or a script file. All error messages begin with the word ERROR.

1.5.5 Radix

ADEX assumes a radix of eight (octal), unless you indicate otherwise. If you wish to change the radix of all numbers to ten (decimal) or sixteen (hexadecimal), use the **RADIX** command. This command also lets you change the radix back to octal. To change the radix of one number, you can use a radix indicator. Follow the number with K for octal, H for hexadecimal, or . for decimal.

1.5.6 Environment Control Word

The Environment Control Word (ECW) is a word in memory that lets you control many aspects of your operating environment. Each bit of the word represents a flag which you can turn on or off, depending on the effect you want. When you bring ADEX up, these flags are set as shown in Table 1-2.

Four commands - **ENVIRONMENT, LOG, NOTIFY,** and **OPERATOR** - let you define the flags in the Environment Control Word. Table 1-2 shows which commands you should use to change the effect of any flag. The same commands let you see the status of the flags that they control. You can also complement (that is, change to the opposite state) the condition of any flag by using a single or double keystroke. These keystrokes provide a short-cut to changing the operating environment.

1.5.6.1 Single Keystroke Commands - You can use single keystroke commands only while a program is running or while something is being printed (that is, when you have no prompt of any kind) to complement the meaning of any flag in the Environment Control Word. Never use these command when you have a CLI prompt or when a program is waiting for input; the commands would be interpreted literally. If you have a prompt, use a double keystroke command.

Single keystroke commands are single digits or alpha characters. Each command corresponds to a flag in the Environment Control Word. For example, to complement the value of the REPORTALL flag, you would type 8. See Table 1-3.

Single keystroke commands are printed as follows:

- On hardcopy terminals and DGC Displays as # plus the character typed.
- On uppercase LCDs, the # plus the character are blinked.
- On upper/lowercase displays, graphics displays, and color displays, the # plus the character are dimmed.

1.5.6.2 Double Keystroke Commands - You can use double keystroke commands whenever you have a CLI prompt or a program is awaiting input. If you were to enter a single keystroke command, the program would interpret it literally. To prevent this from happening, you must precede a single keystroke command with a CTRL-P, thus making it a double keystroke command. For example, to complement the value of the OPERATOR flag, type CTRL-P F.

NOTE

Uppercase or lowercase may be used for the letter commands A-G listed in Table 1-3.

WARNING

DTOS programs may not recognize keystroke commands.

Double keystroke commands are printed as follows:

- On hardcopy terminals and DGC Displays, as ^P# plus the character typed.
- On uppercase LCDs, the ^P# plus the character are blinked.
- On upper/lowercase display, graphics displays, and color displays, the ^P# plus the character are dimmed.

Table 1-2. Environment Control Word Flags

FLAG NAME	EFFECT	INITIAL VALUE	COMMAND
LOOP	Loop when an error is detected.	Yes	ENVIRONMENT
CONSOLE	Print output on system console.	Yes	ENVIRONMENT
PERCENT	Print percentage of failure at end of tests.	No	ENVIRONMENT
PASSES	Notify or start of passes.	Yes	NOTIFY
SECONDARY	Enable output to secondary output device.	No	ENVIRONMENT
PAUSE	Execute the Octal Debugging Tool (ODT) if error is detected.	No	ENVIRONMENT
TESTS	Notify of start of tests.	No	NOTIFY
REPORTALL	Print all errors encountered.	No	ENVIRONMENT
TERMINATE	Terminate program if error is encountered.	No	ENVIRONMENT
LOGGING	Enable console logging.	No	LOG
MODULES	Notify of start of modules.	No	NOTIFY
VERIFY	Run only quick verification tests.	No	ENVIRONMENT
PAGE_MODE	Set page mode (that is, suspend output every 23 lines or when a formfeed is encountered).	No	ENVIRONMENT
ABORT	Terminate current program and abort current script if an error is detected.	No	ENVIRONMENT
OPERATOR	Operator present.	Yes	OPERATOR
DCHANNEL	Initiate background data channel activity on the second and subsequent passes of processor diagnostic test programs via the Data Channel Exerciser Utility. RESTRICTION If the runtime media is a tape, then an I/O tester PCB should be installed (at manufacturing test facility).	No	ENVIRONMENT
MULTIPLE	Allow Diagnostics to run under all processors processors on a multi-CPU machine (MV/20000).	No	ENVIRONMENT

Table 1-3. Single Keystroke Commands

COMMAND	FLAG
1	LOOP
2	CONSOLE
3	PERSENT
4	PASSES
5	SECONDARY
6	PAUSE
7	TESTS
8	REPORTALL
9	TERMINATE
A	ENVIRONMENT
B	MODULES
C	VERIFY
D	PAGE_MODE
E	ABORT
F	OPERATOR
G	ENABLE DATA CHANNEL ACTIVITY
L	MULTIPLE CPUs

1.6 THE FILE SYSTEM

ADEX has a flat file system, which means that all files are on one level. Once the operating system is running, you have access to any file without changing your working environment in any way.

1.6.1 File Types

ADEX has fifteen types of files. These files are summerized in Table 1-4. ADEX includes some system files, such as RSYS, SNRU, SDF, and DRVR. You will probably be aware of the existence of these files, but not of how the files are being used.

System files are required by ADEX in order to run. User files are not. Generally, user files include all help files, test programs, and their overlay and data files.

As you run test programs under ADEX, you may want to be working with PRG, OLF, DTOS, and SRCH files. All four contain executable code.

NOTE

To abort execution of DTOS files, do the following:

CTR-O

@ 77777R

(May not work on all sections of program.)

Table 1-4. File Types

TYPE/ EXTENSION	MEANING
.RSYS	ADEX system programs that become the ADEX operating system when loaded into memory.
.PRG	ADEX utility and diagnostic programs that run under ADEX control.
.DTOS	Diagnostic programs first developed for the DTOS diagnostic operating system and now used in ADEX.
.OLF	Overlay files for the utilities and .PRG diagnostic programs.
.HELP	Files which explain the use and facilities of commands, utilities, and tests.
.UDF	User data files which contain numeric data for diagnostic programs.
.SDF	System data files containing numeric data for ADEX utility programs.
.SNRU	System non-resident utility files. ADEX loads them into memory when they are required.
.DRVR	Driver files, which performs I/O operations to peripheral devices supported by ADEX.
.SOP	Stand-alone diagnostic programs which can be run instead of ADEX.
.SRCE	Sources files which are macro files that contain CLI commands. The .SRCE extension may be substituted with .CLI when referring to macros using the ADEX-CLI>.
.TOGL	Toggle files which you can create to perform simple diagnostic tests which run under ADEX control.
.SRCH	These are overlay files used by the SEARCH.PRG program. They are not found in MV ADEX, which uses the system exerciser program MVSYSTEMX.PRG instead.
.FRU	These are low level diagnostics that report which Field Replaceable Unit(s) is/are the most likely failure(s).
.MIC	Each .FRU has its complementary micro-code (.MIC) file required for operation. System micro-code, used to execute the operating system, is also listed as .MIC.

When you start running macros as a short cut to executing a sequence of CLI commands and test programs, you will be working with SRCE files. You can create source files with either the Script Builder or the Edit utility.

You can also use the Edit utility to create two other kinds of files: TOGL and TXT. Toggle files contain instruction to save you from toggling switches. For example, you could have a toggle file to set up a seeking/recalibrating loop for disks. Text files are simply ASCII text files that can contain any kind of textual information. For example, it can contain a message or a memo.

1.6.2 Filenames

Files are identified by a name and a type. The names of files may be up to 14 characters long and may include any character except a period (.) or a space. When you name a file, follow the name with its type. This extension consists of a period followed by one of the types listed in Table 1-4. The format is:

filename.filetype

For example, RENEE.TXT is a text file, but ANNA_22.SRCE is a source file. A filename followed by an extension is also known as a pathname.

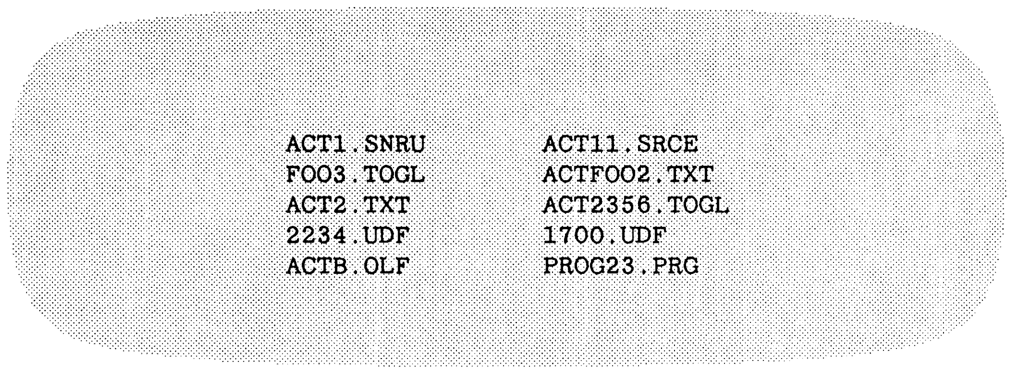
Although not all CLI commands require an extension (that is, the file type), it is suggested that you always provide one.

When naming files, you must identify them uniquely. You can, however, have several files with the same name, but different extensions. For example, you could have files called ANNA.PRG, ANNA.OLF, ANNA.TXT, and ANNA.SRCE.

In working with files, you must refer to them by name. Three CLI commands (**FILESTATUS**, **DELETE**, and **PERMANANCE**) and one ADEX utility (ADEX Media Builder), however, allow you to take a short-cut by using template characters.

1.6.3 Templates

ADEX includes two template characters: * and +. These characters serve as a short-cut in referring to filenames. The * template tells the system to look for any file that includes the characters you indicate, plus any other single character. For example, if you had the files listed in Figure 1-5 and you wanted to find any file beginning with the characters ACT and followed by any other single character, you would use the * template as follows:



```
ACT1.SNRU          ACT11.SRCE
FOO3.TOGL          ACTFOO2.TXT
ACT2.TXT           ACT2356.TOGL
2234.UDF           1700.UDF
ACTB.OLF           PROG23.PRG
```

FS-12831

Figure 1-5. A Selection of Files

ACT*

Given the selection in Figure 1-5, the system would find:

```
ACT1.SNRU          ACT2.TXT          ACTB.OLF
```

Notice that the system finds any file beginning with ACT and followed by any single character, regardless of its filetype extension.

The + template tells the system to look for any file which includes the characters you indicated, plus any number of other characters. For example, to find any file beginning with the characters ACT and followed by any other characters, you would type:

```
ACT+
```

This time the system would respond with (using the files given in Figure 1-5):

```
ACT1.SNRU           ACT11.SRCE          ACTF002.TXT
ACT2.TXT            ACTB.OLF            ACT2356.TOGL
```

You can also use either template to find files which end with the characters you indicate. For example, to find all files ending in 2, you would type:

```
+2
```

Here, because you are looking for all files ending in a certain character, you place the + first. The system would now respond with:

```
ACT2.TXT            ACTF002.TXT
```

Lastly, you can use the templates in the middle of the characters you want to match. For example, if you type:

```
A+1
```

with the selection of files in Figure 1-5, the system would find ACT1.SNRU and ACT11.SRCE.

You can combine the template as follows:

```
*CT+
```

In this case, all the files with one character before CT and with any number of characters after CT will be found. With the files given in Figure 1-5, you would get:

```
ACT1.SNRU           ACT2.TXT            ACTB.OLF
ACT11.SRCE          CTF002.TXT          ACT2356.TOGL
```

1.6.4 Manipulating Files

Several CLI commands allow you to work with the file system. To find out what files are on the runtime media, you can use the **FILESTATUS** command. This command lets you list all files, or just those with a certain extension, and/or those whose names have certain characters. You select the catagories of files by using template characters.

If you are working off a disk, you can find out how much disk space you are using and how much is free with the **SPACE** command.

If you want to see what certain files contain, you can display them with the **TYPE** command. This command, however, applies only to files containing ASCII text (that is SRCE and TXT files).

To make a copy of any file, use the **COPY** command. To add one macro file (SRCE) to another, use the **APPEND** command.

You can also change the name of any file with the **RENAME** command and you can delete any non-permanent file with the **DELETE** command. To make a file either permanent or non-permanent, use the **PERMANENCE** command.

NOTE

The ADEX Media Builder and the Update utility automatically create permanent files. The Edit utility creates non-permanent files.

Two additional commands let you manipulate files: **TLOAD** and **TDUMP. TLOAD** loads a file from the specified file number on the tape drive indicated and **TDUMP** dumps a file to the drive indicated.

1.7 BASIC TROUBLESHOOTING

When troubleshooting a system, you generally do the following:

1. Try to evaluate the situation.
2. Using ADEX, run test programs to isolate the failing device.
3. Repair the system.
4. Verify the repair.
5. Bring down ADEX once you have finished running the diagnostics.

Here, we will discuss Steps 1, 2, 4 and 5. For Step 3, refer to the appropriate maintenance documentation.

1.7.1 Preliminary Evaluation

Whenever possible, talk to the person who was working with the system when the failure occurred. Also, carefully examine any printouts that document the machine's state when it failed.

After gathering all the available information, you should have answers to the following questions:

- Is the failure intermittent or a one-time event?
- Is the system still operational?
- Is the error always the same?
- Do the symptoms form a consistent picture?
- Has any hardware, software, or operating environment been changed recently?
- Has the system ever had similar problems? What were the conclusions at the time?
- What does the user think the problem is?

Your answers to these questions let you form an initial conclusion about the symptoms:

1. They point to a specific device.
2. They point to a general area of the system, but not a specific device.
3. They do not provide a clue to the source of the problem.

1.7.2 Running Test Programs

Once you have an initial conclusion about the problem, you can start running test programs. What test you choose to run depends on your initial diagnosis. If you have no clues about the source of the problem, it is best to use either the RUN or ACCEPT macros. These macros execute a standard group of diagnostic programs on the equipment currently in the System Equipment Table. Be sure, therefore, that the SET is up to date. Also, be sure that any disk or tape drives you are testing are not write-protected.

WARNING

Back-up disk!

The RUN Macro tests each device and function and ensures that each test progam goes through three passes. The ACCEPT macro also tests each device, but has each program make only one pass.

If these tests show nothing, you can try a long run of System Exerciser (SEARCH for NOVA/ECLIPSE ADEX and MVSYSTEMX for MV series. If these still fail to show anything, you should call the Customer Support Center (CSC).

If, on the other hand, you know which device or part of the system is at fault, you should execute the test programs for that device. For details on the test programs, see the individual HELP files (Tape runtime only).

To execute ADEX programs or TOGL files, use the **XEQ** command. The **XEQ** command executes one program at a time (the XEQ is not required on a command line). If you wish to execute several programs in a row, you have two options. You could either set up a script (or macro) file containing a series of commands, or you could use a multiple command line. In a multiple command line, you string the programs together, each separated by a semicolon (;).

If you use a multiple command line, you must be sure that all the names of the programs you want to run will fit on one line (68 characters). If they do not fit on one line, you must write a script. If you want to execute a sequence many times, you should also write a script.

You can execute ADEX programs with the CLI **LOAD** and **START** commands, but this method is not recommended because **LOAD** does not update certain system variables. Therefore you will get erroneous information if these variables are used.

You can also load and optionally start a program on a slave processor (for example, DCU) with the **SLAVE** command.

Depending on the errors the test programs report, follow the procedures in the appropriate maintenance manual.

If you receive any patches (fixes or additions) to test programs, you should update the program files on the ADEX system disk with the Edit utility.

1.7.3 Verifying the Repair

Once you have found the problem in your system and repaired it, verify the repair as follows:

1. Rerun the programs that isolated the problem. If the programs run without errors, the problem has been solved. If the failure reoccurs, restore any FRUs that you changed and start the troubleshooting process again.

2. Run a complete set of system exerciser programs to make sure the system is working properly. If the Exercisers disclose a new problem, start troubleshooting again.

1.7.4 Bringing Down ADEX

Once the diagnostics have finished running, it is necessary to bring down ADEX. To do so, use either **BYE** or **BOOT** commands. Both commands shut down the system in an orderly fashion, but BOOT also lets you reboot the device you choose, such as your normal operating system (like AOS/VS). **BOOT** saves you from using **BYE** and then bringing up the operating system manually. When you use the **BOOT** command, you are able to specify the device you want booted.

SECTION 2
ADEX-CLI

SECTION 2
ADEX-CLI

The ADEX CLI allows you to control a number of system variables, such as the system console, the radix used by the operating system, and the way errors are reported by test programs. The CLI also lets you manipulate the many kinds of files that ADEX supports. In addition, CLI commands let you execute system utilities (such as the Edit utility or Script Builder), as well as test programs.

2.1 USING THE CLI

The CLI can be used in two modes: manual and script. While in manual mode, the CLI processes commands that you enter in response to the CLI prompt, ADEX-CLI>. In script mode, the CLI processes the commands that it finds in a previously created macro file. Macros provide short-cuts to frequently executed sequences of commands.

2.1.1 Command Formats

Certain rules must be observed when entering CLI commands. CLI command lines consist of just a command line alone or a command followed by one or more arguments. An argument is a piece of information the CLI requires to execute the command; for example the name of a file. Some commands require arguments, others allow them, and others do not except them.

When entering CLI commands, uppercase and lowercase letters can be mixed. The CLI does not distinguish between them. When entering numerical information, however, the current radix must be used (octal is the default, but it can be changed). To change the radix of a particular number, add a radix indicator to the number:

K	Octal radix, for example, 7760K
.	Decimal radix, for example, 188349.
H	Hexadecimal radix, for example, 123A9H

If the command line has one or more arguments, the commands and the arguments must be separated with one or more spaces, a tab, or a comma.

For example:

```
ADEX-CLI> ECLASS WARNING
```

or,

```
ADEX-CLI> ECLASS,WARNING
```

Terminate (transmit) each command by pressing the carriage return (CR), NEW LINE, Escape (ESC), or Line Feed (LF) key. The CLI takes no action until one of these keys is pressed. Once, the key is pressed, the CLI tries to interpret the command.

2.1.1.1 Abbreviations – Commands and arguments may be abbreviated as long as the CLI cannot confuse them with any other commands or arguments. The **DELETE** command can be abbreviated as **DEL**, but not as **DE** because that would not distuinguish it from the **DEFAULT** command. When the use of each command is explained, abbreviations are indicated in uppercase letters, for example **DELete**.

2.1.1.2 Multiple-Command Lines – More than one CLI command can be put on a line by separating commands with a semicolon (;). The last command must be terminated by a CR, NEW LINE, ESC, or LF character. Commands are executed in order. For example:

```
ADEX-CLI> LOG;MEDIA
```

The status of logging is displayed first, followed by the current runtime media. If there is a sequence of commands on a line and one contains an error, the CLI processes only those commands (if any) preceding the error (unless the CLI error class is warning or ignore).

NOTE

Only 68 characters fit on one command line.

2.1.2 Screen Edit

On video terminals, the Screen Edit utility can be used to alter, delete, or repeat the command line. Most of the commands this utility accepts consist of control characters. To issue any control character, first press and hold the CTRL key, then press the appropriate character key. For example, to issue a CTRL-U, press and hold CTRL and then press U. If a command key is used incorrectly, or if the command line exceeds 68 characters, a tone will sound to signal that a mistake has been made. Table 2-1 summarizes the Screen Edit control characters.

2.1.3 Making Corrections

On a hardcopy terminal, if an error is made while typing a command line, it can be corrected with either the DEL or RUBOUT key. When one of these keys is pressed, an underscore (_) appears, indicating that the preceding character is being deleted. To delete an entire line, use CTRL-U, which produces a ^U and moves the cursor to the beginning of the next line.

2.1.4 Control Characters

In addition to the Screen Edit control characters, there are other control characters which tell the operating system to take some action, such as interrupting the execution of a program. Table 2-2 summarizes these control characters.

Table 2-1. Screen Edit Commands

CHARACTER	ACTION
CTRL-A	Moves the cursor to the end of the current command line, if the command line above it is shorter. If the command line above is longer, CTRL-A repeats the longer part of the command line on current line. If the cursor is on a blank line, it repeats the entire preceding line. NOTE You cannot repeat a command line after a CTRL-C, CTRL-A or CTRL-C, CTRL-B sequence.
CTRL-B	Positions the cursor at the beginning of the preceding word.
CTRL-E	Starts insert mode. This mode lets you insert characters in the middle of a command line. When you finish inserting, use CTRL-E again to turn off insert mode.
CTRL-H or HOME	Positions the cursor at the beginning of the line.
CTRL-J, CTRL-L, CTRL-M, NEWLINE or CR	Transmits command. Turns off the insert mode (if it was on) and uses the entire line as the command.
CTRL-K or ERASE EOL	Erases all characters to the right of the cursor.
CTRL-U	Deletes the entire line.
CTRL-X or ----->	Moves the cursor one position to the right.
CTRL-Y or <-----	Moves the cursor one position to the left.
ESC	Deletes the entire line, prints $, and issues a new prompt.
DEL or RUBOUT	Deletes the preceding character.

Table 2-2. Control Characters

CHARACTER	ACTION
CTRL-C CTRL-A	Terminates any test program or utility that is currently running. If the CLI is processing a multiple command line or a script file, the program goes on to the next command. If no command follows the one that was terminated, the CLI issues another prompt. When a CTRL-C CTRL-A is issued, the message ABORT appears.
CTRL-C CTRL-B	Aborts a script. The currently running diagnostic is aborted and all other CLI commands are ignored. The message ABORT appears, followed by a CLI prompt. If no script is running, CTRL-C CTRL-B has the same effect as CTRL-C CTRL-A.
CTRL-C CTRL-C	Prints the characters ^C^C if ADEX or a test program is still running. This control character sequence is very useful in checking whether the system is still up or a program has hung. NOTE The response of the system may not be immediate.
CTRL-D <0,...,9>	Complements the corresponding bit (i.e., 0-9) in the 32-bit Switch Register.
CTRL-D <A,...,V>	Complements the corresponding bit (i.e., 10-31) in the 32-bit Switch Register. NOTE To denote a character that is to act as a Switch Register function key, the symbol "&" is printed before the character. For example, typing CTRL-D 8 echos on the system as ^D&8. On video displays, the characters are dimmed or blinking to distinguish them from non-control characters.
CTRL-D CTRL-M	Prints the contents ot the 32-bit Switch Register in the current radix. NOTE CTRL-D is valid in both input mode and polling.
CTRL-O	Disables interrupts if they were on and executes the octal debugger. NOTE To abort execution of DTOS files, do the following: CTRL-O @77777R (May not work on all sections of program.)

Table 2-2. Control Characters (continued)

CHARACTER	ACTION
CTRL-P	Interprets the next character as single keystroke command. If the next character is a control character (i.e., less then 40 octal), do not process the character to see if it is an ADEX control character. Instead, input the character literally. WARNING CTRL-P followed by either NEW LINE, CR, or formfeed is functionally equivalent to ";" when parsed by the CLI.
CTRL-Q	Resumes output to the console (undoes the effect of CTRL-S).
CTRL-S	Suspends output to the console (to resume output, use CTRL-Q).

2.1.5 HELP Command

ADEX CLI includes a **HELP** command, plus a tutorial mode that will provide some assistance when using ADEX. The **HELP** command can be used any time the CLI is in manual mode. The command will provide information about CLI commands, related topics, system programs, and test programs.

NOTE

Tape only!

ADEX CLI also provides tutorial help whenever a system flag called OPERATOR is on and arguments to a command which requires them are omitted. The CLI then steps through a tutorial session to show which arguments must be provided. When ADEX is brought up, OPERATOR is automatically on, so that the tutorial help is readily available unless the flag is turned off.

2.2 CLI COMMANDS

This subsection is a complete reference to all CLI commands. The commands are summarized by function in Table 2-3. Each command is then described in detail. They are listed in alphabetical order for easy reference.

Table 2-3. CLI Commands by Function

COMMAND	FORMAT	MEANING
	Getting Help	
HELP	*[item]*	Displays information about item.
	Working with the System Equipment Table	
EQUIPMENT	*[VERBOSE [device code]]* *[WRITE [filename]]* *[READ [filename]]* *[FLUSH]*	Displays, saves, replaces, or clears the System Equipment Table (SET).
PROTECT	*device_code [unit_number]*	Protects a device against writing.
SCRATCH	*device_code [unit_number]*	Makes the specified disk or tape a SCRATCH media.
UNPROTECT	*device_code [unit_number]*	Removes write protection from a device.
	Controlling the Operating Environment	
CONSOLE	*[device_code [line_number]]*	Displays or sets the system console.
DEFAULT	*[<PASMAX,PORT> [new_value]]*	Displays or sets a system default.
ECLASS	*[ERROR]* *[WARNING]* *[IGNORE]*	Displays or sets the CLI error class.
ENVIRONMENT	*[BRIEF [new_value1] [new_value2]* *[INITIAL]* *[CLEAR,OFF,NO> flag1..flagn]* *[SET,ON,YES> flag1..flagn]*	Displays sets, or clears one or more flags in the Environment Control Word.
INITIAL	*[CPU ID Number]*	Displays the current CPU ID number and the state of the CPUs in the system.
LOG	*[CLEAR[device_code[unit_number]]]* *[START[device_code[unit_number]]]* *[DISPLAY[device_code[unit_number]]]* *[STOP]* *[SIZE[new-size]]*	Displays, enables, or disables console logging; or clears a console log.
NOTIFY	*[<SET,ON,YES><All, Tests, Modules, Passes>]flag1..flagn]*	Displays or defines the level of notification.

Table 2-3. CLI Commands by Function (continued)

COMMAND	FORMAT	MEANING
OPERATOR	*[<SET,ON,YES>]* *[<CLEAR,OFF,NO]*	Displays or defines the status of the operator flag.
RADIX	*[8]* *[10]* *[16]*	Displays or defines the radix.
SECONDARY	*[device_code [line_number]]* *[slave_dev_code mux_dev_code [line_number]]* *[UNDEFINE]*	Displays, defines, or undefines the secondary output device.
STATUS		Displays status of software control flags, runtime media, radix, string, switch register, CLI error class.
SYSID	*[new_system-id]*	Displays or defines the system identification message.
	Manipulating Files	
APPEND	*dest_file source_file*	Appends one script source file to another.
DELETE	*[template]*	Deletes one or more files.
FILESTATUS	*[template]*	Lists all or some files on runtime media.
PERMANENCE	*template [ON]* *[OFF]*	Sets or displays a file's permanent status.
RENAME	*current_name.filetype newname*	Changes a file's name.
SPACE		Displays amount of disk space in use, not usable, and free.
TDUMP	*filename.filetype [device_code[unit [file#]]*	Dumps a file to the tape drive indicated.
TLOAD	*filename.filetype [major_rev [:minor-rev] [device_code [unit [file#]]]*	Loads a file from the specified file number on the drive indicated.
TYPE	*filename.filetype*	Displays a text or script source file.

Table 2-3. CLI Commands by Function (continued)

COMMAND	FORMAT	MEANING
	Working with Media	
MEDIA	*[device_code [unit_number]]*	Displays or sets the runtime media.
PROTECT	*device_code [unit_number]*	Protects a device against writing.
SCRATCH	*device_code [unit_number]*	Makes the specified disk or tape a SCRATCH media.
UNPROTECT	*device_code [unit_number]*	Removes write protection from a device.
ACCEPT*		Runs acceptance tests (one pass per program).
BOOT	*[device_code [unit_number]]*	Shuts down system and boots a device. To boot AOS, you must also set hard console switches (or switch register on soft console) to device code or media you are booting.
BYE	*[NOHALT]*	Shuts down system and optionally restarts CLI.
CHECKSUM	*[pathname]*	Checks the integrity of the resident system code or specified file.
DEFAULT	*[<PASMAX,PORT> [new_value]]*	Displays or sets a system default.
ERMES	*error_code*	Prints an explanation of error.
IAC*		Auto-sequences IAC tests.
ISC*		Auto-sequences ISC tests.
RUN*		Runs standard set of test programs (three passes per program).
SLAVE	*device_code filename [address]*	Loads file into slave processor's memory and optionally starts slave processor.
TESTCOMM*		Auto-sequences the communications diagnostics.

Table 2-3. CLI Commands by Function (continued)

COMMAND	FORMAT	MEANING
[XEQ]	*filename [passes[address]]*	Executes a program.
CHAIN	*script_filename[.srce]* *[macro_arguments]***	Loads in and executes the specified script file.
CLEAR	*STRING* *ALL* *flag1..flagn*	Clears one or more control flags or the string.
CMPSTR	*<R0,R1,R2,R3,WARNING. [message]*	Compares the string on the command line with the current string buffer.
COMMENT	*comments*	Enters a comment.
LABEL	*label_name***	Names a node in a conditional scripts.
RDSCF	*<R0,R1,R2,R3,WARNING. [message]*	Prints the optional message and inputs the new value for the specified software control flag.
RDSWR	*[message]*	Prints the optional message and inputs a string containing one or two 16-bit values to store into SWREG.
READ	*[message]*	Prints an optional message, takes input from the operator, and stores it in the string.
SET	*argument_1 [<+,-,=,&>,argument2]*	Sets some or all of the soft-ware control flags (optionally to value of expression).
STRING	*[new_string]*	Sets or displays the string.
TEST	*flag true_lable [false_label]***	Tests a software flag and proceeds to a node in the script.
WRITE	*[message]*	Prints a blank line or specified message.

Table 2-3. CLI Commands by Function (continued)

COMMAND	FORMAT	MEANING
Communicating with Programs		
STRING	*[new_string]*	Sets or displays the string.
SUBINFO	*filename[.prg]*	Prints the submittal information for the specified ADEX program.
SWREG	*[16_bit-value [16_bit_value]]*	Sets or displays the switch register.

* Denotes macros. They are not really commands.
** Valid only within a script.

ACCEPT

Description

ACCEPT is a macro that executes non-destructive diagnostic programs on all equipment listed in the SET. It operates in an auto-sequence mode, with no questions asked. You may modify this run list from disk run-time media using the following procedure:

1. Xeq SCRBLD and Save (S) the new source file (NEWMACRO.SRCE).

2. Edit this new source file using the Edit Utility (Sec 3).
   For example: You may want to delete the IAC tests from a system with a large IAC configuration.

NOTE

Some equipment cannot be tested in auto-sequence script because of program restrictions.

APPEND

Syntax: **APpend** *dest_filename[.SRCE] source_filename[.SRCE]*

dest_filename is the name of the source file to which you are appending another. *source_filename* is the name of the source file you are appending. You can optionally include the .SRCE extension.

Description

APPEND appends one source file to another.

Example

ADEX-CLI> **AP** *RICH.SRCE ANNA*

Appends a source file called *ANNA* to one called *RICH.*

BOOT

Syntax: **BOot** *[device_code [unit_number]]*

device_code is the device you wish to boot. If you do not provide a device code, the CLI assumes that you want to bootstrap the runtime media. The *unit_number* is the number of the device. If the unit number is omitted, unit 0 is assumed.

Description

BOOT shuts down the operating system in an orderly fashion and boots the specified device.

CAUTION

> In order to boot AOS from ADEX, you must set the hard console switches (or the switch register in a soft console) to the device code of the AOS media you are booting.

Examples

ADEX-CLI> **BOOT** *27 1*

Boots unit 1 on device code 27.

ADEX-CLI> **BOOT**

Boots the system media.

BYE

Syntax: **BYe** *[Nohalt]*

Selecting the *Nohalt* option allows you to get back to ADEX CLI by pressing NEW LINE. (If *Nohalt* is not selected, the processor is halted.)

Description

BYE shuts down the system. The console log is closed. The runtime media is rewound/recalibrated or returns to previous menu.

Example

```
ADEX-CLI> BYe

You cannot continue execution.
```

CHAIN

Syntax: **CHAin** *script_filename[.srce] [macro_arguments]*

Description

CHAIN loads and executes the specified CLI source file. If the file type extension is excluded, file type .SRCE is assumed. **CHAIN** is used in a macro to explicitly direct the sequencing of macros. (This command is valid only in Script Mode.)

Example

```
ADEX-CLI>          TYPE MACRO.SRCE
                   WRITE HELLO
                   MACROA
                   WRITE GOODBYE.

ADEX-CLI>          TYPE MACROA.SRCE
                   WRITE HOW ARE YOU
                   CHAIN MACROB
                   WRITE THIS DOES NOT PRINT

ADEX-CLI>          TYPE MACROB.SRCE
                   WRITE FINE

ADEX-CLI>          MACROB

                   HELLO
                   HOW ARE YOU
                   FINE
                   GOODBYE.
```

CHECKSUM

Syntax: **CHEcksum** *[pathname]*

Description

CHECKSUM verifies the integrity of the memory resident operating system code by summing all the memory resident code and comparing the total to a known value. When used with the optional argument pathname, **CHECKSUM** performs a checksum on the specified file. If a FILE CHECKSUM ERROR is returned, then the checksum word for the file will have been updated to reflect the new contents of the file.

WARNING

This commands only catches single bit errors.

Example

ADEX-CLI> **CHE**

Performs an immediate checksum on the resident operating system.

CLEAR

Syntax: **CLear** *[String]*
[All]
<R0,R1,R3,Warning,Error>

Description

CLEAR is a conditional CLI command. Depending on the argument, **CLEAR** clears the string buffer (i.e., sets the current string to null), clears the specified software control flags (i.e., R0, R1, R2, R3, Warning and/or Error), or clears all the software control flags.

Example

ADEX-CLI> **CL** *R2 Error*

The *R2* and *ERROR* flags are cleared.

CMPSTR

Syntax: **CMpstr** *<R1,R2,R3,Warning,Error> [string]*

Description

CMPSTR compares the string on the command line with the current string buffer. If the two strings match, the specified software control flag is set to true (=+1). Otherwise, the software control flag is set to false (=0).

If string is not supplied, **CMPSTR** sets the specified software control flag to true if the first byte of the current string buffer is null. Otherwise, the software control flag is set to false.

CMPSTR can be executed only from a script. If you execute the command in manual mode, ADEX returns a COMMAND NOT VALID IN MANUAL MODE warning.

If no arguments are supplied, ADEX returns a COMMAND REQUIRES ARGUMENTS warning.

Example

ADEX-CLI> **CM** *R1 ERCC ERROR*

Compares *ERCC ERROR* with the current string buffer.

COMMENT

Syntax: **COMment** *[anything useful]*

Description

When writing scripts, **COMMENT** allows you to insert the comments of your choice into a CLI macro. If console logging is enabled, **COMMENT** lets you enter comments into the console log.

Example

ADEX-CLI> **COM** *Problem occurred of 1-10-82.*

Assuming console logging is enabled, the comment *Problem occurred on 1-10-82.* is entered into the console log.

CONSOLE

Syntax: **CONsole** *[slave_device_code [line_#]]]*

Description

When used alone, **CONSOLE** displays the current system console type (VIDEO or HARDCOPY), model number, device code, (input and output if the device is a TTI/TTO), and if the device is a MUX, the line number and its characteristics. (Console type, model number, and line characteristics information is derived from the SET.) If the system console is on a slave processor bus, the device code of the slave processor is printed in parenthesis after the multiplexor's device code.

With the argument *device_code*, **CONSOLE** changes the system console to the specified device_code. The console type is set according to the SET. The device code specified refers to the input device (e.g., CONSOLE 10).

With the arguments *device_code line_#*, **CONSOLE** changes the system console to the device code and line number specified.

With the arguments *slave_device_code device_code line_#*, **CONSOLE** changes the system console to the line, device code, and slave device code specified.

NOTE

The device codes are parsed according to the current radix. The line number is always parsed in decimal.

Example

ADEX-CLI> **CON** *65 34 1*

Switches the console to the terminal on line 1, MUX 34, and IOP device code 65.

COPY

Syntax: **COPy** *destination_pathname srce_pathname*

The original file is *srce_pathname* and the new file is *destination_pathname*. Copies *srce* to destination.

Description

COPY copies the original file into a new file. The new file must not already exist. The file type of the new file does not have to match that of the original file.

Example

ADEX-CLI> **COP** *book.txt preface.togl*

Copies the file preface to the file book.

DEBUG

Syntax: **DEBug** *filename[.prg] [start_address]*

Description

First, **DEBUG** loads the ADEX Debugger and then the ADEX program, starting at location 0. Next, it transfers control to the debugger.

NOTE

A warning is issued if there is not enough memory to hold both the ADEX Debugger and program, or if you provide an illegal filetype extension (it must be .PRG).

If you provide only a filename, this command loads the debugger at the top of user space, loads the ADEX program at location 0, prints the start address of the debugger, and starts the debugger.

If you add the *start_address* argument, the command loads the debugger at the address specified, then loads the ADEX program at location 0, prints the start address of the debugger, and starts the debugger.

Example

ADEX-CLI> **DEB** *INT.PRG*

Loads the debugger at top of user space, loads program INT at location 0, prints start address of the debugger, and starts it.

DEFAULT

Syntax: **DEFault** *[PASMAX,PORT>[new_value]]*

Description

DEFAULT displays or sets system default values. When used alone, **DEFAULT** displays the current system default values: *PASMAX* or *PORT*. *PASMAX* is the number of passes a test program makes when executed via an XEQ filename command. A *PASMAX* of −1 implies "loop forever". *PORT* is the default user port (I/O channel). The maximum valid port number is 6.

If you use **DEFAULT** with either *PASMAX* or *PORT*, it displays the specified default value. If you provide a *new_value*, **DEFAULT** sets the specified value to the *new_value*.

Example

ADEX-CLI> **DEF** *PASMAX 2*

Sets the default value at *2*. Programs make two passes if the pass count on the **XEQ** command is omitted.

DELETE

Syntax: **DELete** *template[.filetype]*

Description

DELETE eliminates file(s) specified from disk runtime media. You can use templates to indicate which files to delete. The system requires you to confirm if you are in manual mode. If the command is executed from a macro, you do not have to confirm.

Example

ADEX-CLI> **DEL+.***srce*

Deletes all source files from disk runtime media.

NOTE

You cannot delete a permanent file. Use the **PERMANANCE** CLI command to change permanence in order to delete files.

ECLASS

Syntax: **EClass** *[Ignore,Warning,Error]*

Description

ECLASS sets or displays the current CLI error class, which determines whether or not a script or long command line will be continued if an error is detected. The error class is initially set to *ERROR*.

If the error class is *IGNORE*, the script or command line will continue despite the error and you will not see any error message. If the CLI error class is set to *WARNING*, the script or command line will continue, but a WARNING message will be printed. Setting the class to *ERROR* will cause the script or command line to be aborted and an ERROR message is printed.

Example

```
ADEX-CLI> EC
```

Displays the current error class. You can also use the STATUS command to display the error class.

EDIT

Syntax: **EDit** *filename.filetype*

Description

EDIT is used to edit a specified file. Data in source or text files are represented as ASCII characters. In all other file types, data is represented as numbers according to the current radix.

A warning is given if:

- The file does not exist and ADEX could not create a new one. The message, FILE CREATION ERROR is printed with an explanation. Execution continues in EDIT–CLI, but the edits cannot be saved.
- The EDIT command was used in a macro.
- The OPERATOR flag is off.

If you do not provide a file type extension, the question FILE TYPE? will appear. You can answer with any one of the fifteen file types. You have the choice of creating a new file if you specify a SRCE, TXT, or TOGL file and there is enough space on the runtime media and in the directory.

WARNING

If a file checksum error is encountered while reading in the edit file, then the session is aborted if the CLI error class is ERROR. If the error class is WARNING, the system prints a warning message but continues to execute the edit utility.

Example

ADEX–CLI> **ED** *ART.TXT*

Allows you to edit the text file, *ART*.

ENVIRONMENT

Syntax: **ENvironment** *[Brief[new_value1][new_value2]]*
[Initial]
[<Clear,Off,No>[flags]]
[<Set,On,Yes>[flags]]

flags is a list of one or more Environment Control Word (ECW) flags.

Description

ENVIRONMENT allows you to modify and/or display the current operating environment, which is controlled by flags contained in the Environment Control Word. This command lets you define the value in the following flags:

- LOOP
- CONSOLE
- PERCENT
- SECONDARY
- PAUSE
- REPORTALL
- TERMINATE
- VERIFY
- PAGE_MODE
- ABORT
- DCHANNEL
- MULTIPLE CPUs

Without an argument, **ENVIRONMENT** prints the current operating environment, including the appropriate single keystroke command and the value of the ECW.

With the argument *Brief*, it prints the operating environment, but only includes the value of the ECW. The contents of the ECW are set according to the *new_value*. *new_value1* sets the contents of bits 0–15 in the ECW. If you add *new_value2*, bits 16–31 in the ECW are also set. In either case, the value of bit 10 (console logging) is not affected.

With the argument *Initial*, it restores all flags to their original value. You can clear the indicated flags with the arguments *Clear, Off, No,* and set the indicated flags with the arguments *Set, On, Yes.*

NOTE

In addition to using Environment to define the flag in the Environment Control Word, you can change the value of a flag by using single or double keystroke commands.

Example

ADEX-CLI> **EN** *Clears Secondary Loop*

Clears the secondary and loop flags.

EQUIPMENT

Syntax: **EQuipment** *[Verbose [device_code]]*
[Write [filename[.udf]]]
[Read [filename[.udf]]]
[Flush]

Description

EQUIPMENT allows you to save, display, write over (modify), or clear the contents of the SET. Without arguments, this command displays the contents of the SET, excluding device dependent information.

Verbose adds the display of device dependent information, i.e., model numbers.

device_code lets you display the information for whatever device you specify. For devices without a code, such as, CHAR, use device code 77 (octal).

Write is used to update the file EQUIP0, which contains the contents of the current SET. This file is created if it does not exist. If you add a filename, the contents are written to the file specified. This must be a UDF type file.

NOTE

Disk runtime only.

Read replaces the current SET with the contents of EQUIP0. Adding a filename to EQ READ replaces the SET with the contents of the file you specify. The file must be a UDF.

Flush clears the contents of the SET so that a new SET can be generated manually.

Example

ADEX-CLI> **EQ** *W*

Writes contents of the SET into the file EQUIP0. It creates the file if it doesn't already exist.

ERMES

Syntax: **ERmes** *error_code*

Description

ERMES prints an explanation of the specified error code.

Example

ADEX-CLI> **ER** *23*

Explains error code *23*.

FILESTATUS

Syntax: **Filestatus** *[template [.filetype]]*

Description

FILESTATUS lists the file type, file name, revision number (major only), and size (number of 256-word pages) of the specified files on the runtime media. You can provide a template character (a * or a +), as well as a filename or a file type to specify certain files you want to see listed.

Examples

ADEX-CLI> **F** +*.srce*

Lists all the source files on the runtime media.

ADEX-CLI> **F** *tes**

Lists all the files beginning with *tes* plus one other character.

HELP

Syntax: **Help** *[item]*

Description

HELP provides information on a specified item (topics, CLI commands, and programs). If you do not provide an argument, a help file is displayed to provide you with information on general operating procedures and how to get further help. Help files for programs include the part number, a description, and the definition of any switch register bits the program uses.

Example

ADEX-CLI> **H** *OPERATOR*

Gives you information on the **OPERATOR** command.

IAC

Description

The IAC macro auto-sequences all twelve IAC tests.

NOTE

IAC must be installed on your system in order to run the IAC tests. IAC will appear on the autosizer listing if it has already been installed on your system.

INITIAL

Syntax: **INitial** *[CPU ID Number] <NL>*

Description

Alone, **INITIAL** displays the current CPU ID number and the state of all CPUs in the system. With an argument, INITIAL starts the requested CPU (if it exists) and then reports the state of all CPUs in the system.

Example

```
ADEX-CLI> INITIAL

                CPU0  IS RUNNING
                SYSTEM MICROCODE IS LOADED

                CPU1  IS STOPPED
                SYSTEM MICROCODE IS NOT LOADED
```

ISC

Description

The ISC macro auto–sequences the three ISC tests.

NOTE

ISC must be installed on your system in order to run the ISC tests. ISC will appear on the autosizer listing if it has already been installed on your system.

LABEL

Syntax: **LAbel** *label_name*

The *label_name* is 1 to 5 contiguous alphanumeric characters, including all punctuation characters, except semicolon (;).

Description

LABEL is a conditional CLI command used only in scripts. When used with the **TEST** command, **LABEL** marks the point where the CLI will continue to execute commands. The commands following the **LABEL** command are executed if the control flag last executed transferred control to the label name. If no control flag is currently being tested, the commands are also executed. **LABEL** may not be used in the manual mode.

Example

```
ADEX-CLI>          TYPE LABEL_EX.SRCE
                   SET R0
                   TEST R0 TRUE
                   WR THIS IS NOT PERMITTED
                   TEST R0 NEXT NEXT
                   LABEL TRUE
                   WR R0 IS TRUE.
                   LABEL NEXT

ADEX-CLI>          LABEL_EX R0 IS TRUE
ADEX-CLI>
```

LOG

Syntax: **LOG** *[STOp]*
[Clear [device_code] [unit]]
[STArt [device_code] [unit]]
[Display [device_code] [unit]]
[Size [new_size]]

Description

LOG lets you save all output sent to the system console or the secondary output device. Console logging is enabled when output is being copied to the Console Log. It is disabled when output is not being copied. (When ADEX is first brought up, console logging is disabled.)

On disk media, the Console Log is saved in a disk file called CONLOG (file type RSYS). Alternatively, the Console Log can be written out to a log tape. The runtime media in this case can be either tape or disk.

LOG, without arguments, prints the current status of console logging, and if enabled, to what device code and unit number.

With the argument *Clear*, **LOG** clears all entries from the runtime media console log file. *Clear device_code* clears all entries from the log tape on the specified device. If you add the unit argument, all entries are cleared from the log tape on the specified device and unit number. If the unit is not specified, 0 is assumed.

With the argument *Start*, **LOG** starts or continues console logging to the disk file CONLOG. A console log entry is made to record that console logging was started. This sets the ENABLE CONSOLE LOGGING flag in the Environment Control Word. *Start device_code* starts or continues console logging to the tape drive located on the specified device code, unit 0. *Start device_code* unit starts or continues logging to the tape drive located on the specified device code and the specified unit.

LOG *STOP* disables console logging and clears the flag in the Environment Control Word. A console log entry records that console logging was stopped. A warning is issued if logging was already disabled.

LOG *display* prints the disk console log stored in CONLOG. The runtime media must be a disk. You will receive a warning if it is not a disk. The console log cannot be displayed if console logging is enabled. With *display device_code*, LOG prints the console log found on the specified tape drive, unit 0. With *display device_code* unit, **LOG** prints the console log found on the specified tape drive and the specified unit.

LOG *size* displays the current size of the console log file (CONLOG.RSYS). This command is only valid when the runtime media is a disk.

LOG *size new_size* changes the size of the disk console log file to the size specified. *New_size* represents the number of sectors to make CONLOG.RSYS, and is interpreted in decimal (unless a radix indicator is used). Changing the console log size has the following requirements and effects:

1. The *new_size* must be between 8 and 32767. Otherwise, an INVALID ARGUMENTS error is returned, and no action is taken.

2. The *new_size* must be large enough to accommodate all the currently used sectors in the console log plus one. Otherwise, an ILLEGAL FILE SIZE error is returned, and no action is taken.

3. If the *new_size* satisfies requirements 1 and 2, and a CONSOLE LOG OVERFLOW condition was present, then console logging is restartable via a **LOG** start command (without having to perform a LOG *clear*) when the **LOG** *size new_size* command is finished.

4. The console log size should be returned to 8 sectors before doing any media builds. If it is not returned to 8 sectors, CONLOG.RSYS will not be transferred to the build media because it is a contiguous file which exceeds 2 KW. If you get an ILLEGAL FILE SIZE error when you try to reduce the log size to 8 sectors, you must first clear the console log, so that it is large enough to accommodate all the currently used sectors plus one.

Example

ADEX-CLI> **LOG** *Start 22*

Starts or continues console logging to the tape on device code 22, unit 0.

MEDIA

Syntax: **MEdia** *[device code [unit number]]*

Description

Without an argument, **MEDIA** displays the runtime media. The display includes the model number, device code, and unit number.

With an argument, **MEDIA** lets you set the runtime media to the specified device code and/or unit number. Console logging is disabled by the MEDIA command (when used with an argument). The media device specified must be listed in the SET as ADEX SYSTEM MEDIA. If it is not, a warning is issued and the original media is maintained.

Example

```
ADEX-CLI> ME 24
```

Changes the runtime media to device code 24, unit 0.

NOTIFY

Syntax: **Notify** *[<Clear,Off,No> [All]]*
<Modules,Passes>]
[<Set,On,Yes> [ALL]]
<Tests,Modules,Passes>]

Description

NOTIFY displays or defines the level of notification. The operator can be notified of the start of tests or modules and/or the end of passes.

Without an argument, **NOTIFY** displays whether the operator is being notified of any of the above.

Set *All, On All,* or *Yes All* sets the notification level to the start of tests, the start of modules, and the end of passes. *Clear All, Off All,* or *No All* clears the notification level so that no messages are printed. You can *Set* the value of the flag in the Environment Control Word by specifying which level you want set (*Tests, Modules,* or *Passes*). You can also *Clear* the specified flag in the ECW.

NOTE

You have the option of using single or double keystrokes to complement the value of the following flags in the Environment Control Word:

- Passes
- Tests
- Modules

Example

ADEX-CLI> N *Set Tests*

Sets the flag in the ECW to YES. You will be notified at the start of the tests.

OPERATOR

Syntax: **OPerator** *[<ON,SET,YES>]*
[<OFF,CLEAR,NO>]

Description

OPERATOR displays or sets the status of the **OPERATOR** flag in the Environment Control Word. If the flag is on, the programs prompt you for input. In the CLI, you have the advantage of being in tutorial mode when the flag is on. If you forget to provide arguments to a command which requires them, the CLI takes you through a brief tutorial session. If the OPERATOR flag is off, you cannot give any program data.

Without an argument, the CLI indicates whether the OPERATOR is on or off. With an argument, the CLI sets the OPERATOR to the state you request.

NOTE

You have the option of avoiding the **OPERATOR** command if you use keystroke commands.

Example

ADEX-CLI> **OP** *OFF*

Turns off the OPERATOR flag.

NOTE

With the OPERATOR flag off, you can execute auto-mode versions of functional Diagnostics or Exercisers.

PERMANENCE

Syntax: **PErmanence** *template[.filetype] [ON,OFF]*
filename[.filetype] [ON,OFF]

Description

PERMANENCE displays or sets the permanence of the specified file(s). Permanent files cannot be deleted. Non-permanent files can be deleted. To display the permanence of a file, use the command along with the appropriate filename, filetype, and/or template. To set permanence on one or more files, add the *ON* argument. To disable permanence on one or more files, add the *OFF* argument.

Example

ADEX-CLI> **PE** *FERG.TXT ON*

Sets the permanence of the text file *FERG* to *ON*.

PROTECT

Syntax: **PRotect** *device_code [unit_number]*

Description

PROTECT enables write–protection on the specified disk or tape on the device code specified. Unit 0 is assumed if no other number is named. If you issue the command in manual mode, ADEX requests a conformation. ADEX verifies that the device specified is write–protected.

Example

ADEX–CLI> **PR** *33*

Assuming that you issued this command in a script, ADEX makes device 33 write–protected and prints the message: DEVICE 33,0 IS NOW WRITE PROTECTED.

RADIX

Syntax: **RAdix** *[8]*
[10]
[16]

Description

RADIX displays or sets the numbering system used for all numeric input and output. Certain values cannot be changed. TEST, MODULE, PASS, and line numbers are always decimal. Device codes are always octal. The radix displayed or entered is in decimal. The default radix is octal.

Example

ADEX-CLI> **RA** *10*

Sets the numbering system to decimal.

RDSCF

Syntax: **RDSCf** *<R0,R1,R2,R3,WARNING> [message]*

Description

RDSCF prints the optional message and inputs the new value for the specified software control flag. A warning is issued is the OPERATOR flag is off.

Example

ADEX-CLI> **RDSCF** *R1 New R1 value?*

Prints the message *New R1 value?* and inputs the new value for R1.

RDSWR

Syntax: **RDSWr** *[message]*

Description

RDSWR prints the optional message and inputs a string containing one or two 16-bit values to store into SWREG. If one value is entered, it is stored in SWREG, bits 0–15. If two values are entered, the second value is stored in SWREG, bits 16–31,

A warning is issued if the OPERATOR flag is off.

Example

ADEX-CLI> **RDSW** N*ew SWREG value?*

Prints the message *New SWREG value?* and waits for the operator to input SWREG values and press NEW LINE.

READ

Syntax: **REAd** *[message]*

Description

READ prints message and inputs string (up to 80 characters long) from the operator. The OPERATOR flag must be on or a warning is issued. This command is useful in macros when you want to pause to wait for the operator to do something.

Example

```
ADEX-CLI> COMMENT Example assumes device code 22,0 is listed as SCRATCH in SET.
          OPERATOR ON
          COMMENT Command not necessary if OPERATOR flag is already on.
          READ Hit NEW LINE when scratch tape is mounted on mag tape unit 0.
          XEQ MTA0_DUMP
```

This example shows you a portion of a script file. **READ** puts a pause in the macro so the operator can, in this case, mount the scratch tape.

RENAME

Syntax: **REName** *current_filename.filetype new_filename[.filetype]*

Description

RENAME changes the name of a file. The new file cannot already exist. You must specify the type of the current file. If you specify a filetype for *new_filename*, it must match the filetype specified for the *current_filetype*.

Example

ADEX-CLI> **REN** *greg.txt hughes*

Renames the current text file *greg* to the new text file *hughes*.

RUN

Description

RUN is a macro that tests each device, option, and function of the current system configuration. It runs each program for three passes.

NOTE

Some equipment cannnot be tested in auto-sequencing script because of program restrictions.

SCRATCH

Syntax: **SCratch** *device_code [unit_number]*

Description

SCRATCH makes the specified disk or tape a SCRATCH media. Unit 0 is assumed if none is specified. If the command is executed in manual mode, ADEX requests a confirmation before the device is made a SCRATCH media.

After write-enabling the device, ADEX verifies the new device status with the message DEVICE *dc,un* is now a SCRATCH media.

Example

ADEX-CLI> **SCRATCH** *22,1*

Makes device code 22, unit 1, a SCRATCH media.

SECONDARY

Syntax: **SECondary** *[slave_device_code[device_code[line_#]]]*
[Undefine]

Description

SECONDARY defines, undefines, or displays the secondary output device. When you define a new device code for the secondary ouput device, it must not be the same as the current system console output device code.

Alone, **SECONDARY** displays the secondary output device.

With the argument *device_code,* **SECONDARY** defines the specified device code to be the secondary output device. If you add *line_#,* the command defines the specified line number of the specified device code to be the secondary output device.

With all three agruments *(slave_device_code, device_code,* and *line #)*, the command changes the secondary output device to the line, device code, and slave device code specified.

With the argument *undefine* **SECONDARY** undefines the secondary output device.

Example

ADEX-CLI> **SEC** *Undefine*
ADEX-CLI> **CONSOLE** *sodc*
ADEX-CLI> **SECONDARY** *condc*

The system console was on device code condc. The secondary device code was *sodc*. The two have now switched device codes.

WARNING

To start printing on the secondary output device, you must set the **SECONDARY** flag in the Environment Control Word, using the **ENVIRONMENT** command.

SET

Syntax: **SET** *argument_1 [[+,-,=,&] argument_2]*

argument_1 is defined as *[All,< R1,R2,R3,WARNING>]* and
argument_2 is defined as *[[R0,R1,R2,R3,WARNING],Number]*

Description

SET *ALL* sets R0, R1, R2, R3, and the Warning software control flags to true, which is defined as non-zero. The actual value is +1. If you use **SET** with any one of the flag names as an *argument,* it sets just that flag to +1.

By providing an *argument_1* + *argument_2,* you add the value of *argument_2* to the software control flags in *argument_1.* The same applies to *argument_1* - *argument_2,* only this time you subtract the values. With *argument_1* = *argument_2,* you set the software control flags listed in *argument_1* to the value listed in *argument_2.* For each of these options, if *argument_2* is another software control flag, then the value of that flag is added, subtracted, or equalled.

By providing an *argument_1* & *argument_2,* you set the software control flag(s) in *argument_1* equal to the logical AND of the value of the software control flag with the value in *argument_2.* If *argument_2* is another software control flag, then the value of that flag is ANDed with the flag(s) listed in *argument_1.* If *argument_2* is a number, then it is ANDed with the flag(s) listed in *argument_1.*

NOTE

You cannot manually set the Error software flag.

Example

ADEX-CLI> **SET** *ALL* + *R0*

Adds the value of R0 to all software control flags.

ADEX-CLI> **SET** *R1 R3-5*

Subtracts 5 from the software control flags R1 and R3.

ADEX-CLI> **SET** *R0* + *R0*

Multiplies R0 by 2.

SLAVE

Syntax: **SLave** *device_code filename{.filetype] [start_address]*

Description

SLAVE loads the specified file into the slave processor on the specified device code . The device is started at address supplied. It is not started if an address is not given. If you do not supply a file type, ADEX looks under type .PRG only. A slave processor that is supported by ADEX is IOP.

If the file loaded into the slave processor is an ADEX program, the **SLAVE** command also loads the appropriate support routines into the processor and handles all system console input/output to and from the slave processor until the program completes.

Example

ADEX-CLI> **SLAVE** *65 IOP_MON.PRG 200*

Loads *IOP_MON.PRG* into the IOP on device code 65 and starts the IOP at location 200.

SPACE

Syntax: **SPace**

Description

SPACE lists, in decimal, the number of disk sectors currently free, unusable, and already used by files.

Example

ADEX-CLI>**SP**

Results in the following:

```
---CURRENT DISK SPACE (SECTORS)---

          FREE= xxxxx
          UNUSABLE= xxxxx
          USED= xxxxx
```

STATUS

Syntax: **STATus**

Description

STATUS prints information about the following variables:

- The Software Control Flags (R0–R3, Error, and Warning)o
- The Runtime Media
- The Current Radix
- The Current Switch Register (32 bits)
- The Current String
- The Current CLI Error Class

Example

ADEX-CLI> **STAT**

```
          ---CURRENT SYSTEM STATUS--

SOFTWARE CONTROL FLAGS:              RUNTIME MEDIA:
   RO= 1                                MODEL NUMBER= 6214
   Rl= 0                                DEVICE CODE= 27
   R2= 0                                UNIT NUMBER= 0
   R3= 0
   WARNING= 2                           SWRWG= 000000 000000
   ERROR= 0                             RADIX= 8.

CLI ERROR CLASS= ERROR
```

STRING

Syntax: **STRing** *[new_string]*

Description

STRING sets or displays the current string buffer (holds up to 80 characters).

Example

ADEX-CLI> **STRING** *MVSYSTEMX.OLF 14*

Inserts the string *MVSYSTEMX.OLF 14* into the string buffer for use by the program TEST_PROG.

SUBINFO

Syntax: **SUbinfo** *filename[.prg]*

Description

SUBINFO prints the submittal information for the specified ADEX program. This submittal information consists of the program name, part number, major revision level, minor revision level, and submittal.

All submittal information (except the minor revision level) is obtained by examining the Submittal Information Block that resides in locations 202 to 217 (octal) in the specified ADEX program.

Example

```
ADEX-CLI> SUBINFO PRIOR_X

          Filename:        PRIOR_X
          Part Number:     096-002500
          Revision:        6.2
          Submittal:       3

ADEX-CLI>
```

The operator has requested submittal information on *PRIOR_X.PRG*. ADEX confirms the existance of *PRIOR_X.PRG* and gives the part number for its listing, 096-002500. The revision level is 6.2, which means that the major revision level is 6, and that Patch #1 and Patch #2 have been installed. (A minor revision level of 0 would mean that no patches have been installed.) The submittal is 3, which means that it is the fourth (i.e., 0-3) submittal of P*RIOR_X*.

SWREG

Syntax: **SWReg** *[16_bit_value [16_bit_value]]*

Description

SWREG sets or displays two 16-bit switch registers which is accessible to all diagnostic programs. The register is used to pass numerical information to a diagnostic program or ADEX utility.

CAUTION

Do not confuse this with the Environment Control Word which controls the operating environment.

Without an argument, the current contents of the switch register are displayed. To store a specified value in the switch register, supply the 16_bit_value. To store two separate values in SWREG, supply both values.

Example

ADEX-CLI> **SWR** *16_bit_value*

Places the *16_bit_value* in bits 0-15 of the switch register.

NOTE

For more details on the SWREG values, see subsection 3.1.1.

SYSID

Syntax: **SYsid** *[new_system_id]*

Description

SYSID displays or sets the system identification message. You can use the message to note the model number, build date and time, system identification number, and owner.

Example

ADEX-CLI> **SYSID** *ECLIPSE S/120 ADEX DIAGNOSTIC MEDIA;* **COMMENT** *NEXT COMMAND*

Changes the system identification to reflect the contents of the ADEX system disk. Note that the **COMMENT** command (in addition to the command terminating character ";") does not become part of the new system identification.

TDUMP

Syntax: **TDump** *filename.filetype [device_code [unit [file#]]]*

Description

TDUMP dumps *filename.filetype* to the magnetic tape drive specified. The default values are device code 22, unit 0, and file 0. The tape drive specified must be listed in the System Equipment Table as SCRATCH.

After the file is dumped, the file name and the file type are printed as a verification that the dump completed successfully.

To load the file from the dump tape to an AOS file, use the **COPY** command, specifying an input magnetic tape record size of 8192 bytes (e.g., **COPY/IMTRSIZE=8192** MYFILE @MTA0:0).

Example

ADEX-CLI> **TDUMP** *DATA.UDF 22,0,3*

Dumps the file *DATA.UDF* to file 3, unit 0 of device 22.

TEST

Syntax: **TESt** *[R0,R1,R2,R3,Warning,Error}* *true_label* *[false_label]*

Description

TEST is used in a script to test the value of one of the six software control flags. If the value of the flag is true, then the CLI continues executing commands after the specified *true_label*. If the value is false, CLI execution continues after the *false_label*. If the flag is false but no label is provided, the CLI will continue executing at the next statement.

Example

```
WRITE    **TESTING ECLIPSE STANDARD INSTRUCTION SET**
         X EASIS_X1
         X EASIS_X2
         X EASIS_X
         3 X EASIS X4
         TEST ERROR YES
         CLEAR R0;WRITE **TESTS PASSED**;TEST R0 EXIT
         LABEL YES; SET R0 WRITE **TESTS FAILED**
         LABEL EXIT
```

The **TEST** command is used here to test the error flag. If the flag is set to true, execution continues at *LABEL YES*. If it is set to false, execution continues at the next line.

TESTCOMM

Description

The **TESTCOMM** macro auto-sequences the communications diagnostics. Edit the file per your configuration.

TLOAD

Syntax: **TLoad** *filename.filetype [major_rev[minor_rev] [device_code [unit [file#]]]*

Description

TLOAD loads the file from the specified file number on the magnetic tape drive indicated. The default values are major revision level 0, minor revision level 0, device code 22, unit 0, and file 0. The tape drive specified must be listed in the System Equipment Table as SCRATCH, WRITE PROTECTED, or UPDATE.

The file will not be loaded onto the runtime media if a file of the same pathname already exists on the runtime media as a permanent file (a CANNOT DELETE PERMANENT FILE error is returned).

The load file on the tape must be in AOS format. Under AOS, use the **COPY** command to dump the file to the tape (e.g., **COPY** @MTA0:0*filename*). After the file is loaded, the file name and the file type are printed as a verification that the load completely successfully.

Files loaded by **TLOAD** are created as non-permanent files (regardless of whether the file existed prior to the **TLOAD**).

If a device status error is detected by TLOAD, the device code, unit number, and DIA status register of the tape is printed.

WARNING

TLOAD is a simpler alternative to the Update Utility. However, if you prefer the safety features that the Update Utility contains, do not use the **TLOAD** command.

Example

ADEX-CLI> **TLOAD** *PRIOR_X.PRG 6:1 22,1,2*

Loads the file *PRIOR_X.PRG* (with a major revision level of 6, and a minor revision level of 1) from file 2, unit 1, of device 22.

TYPE

Syntax: **TYpe** *filename[.srce,.txt]*

Description

TYPE types out the specified source or text file.

Example

ADEX-CLI> **TY** *WEASEL.TXT*

Types out the text file *WEASEL*.

UNPROTECT

Syntax: **Unprotect** *device_code [unit number]*

Description

UNPROTECT changes the status of the specified device code and/or unit number to write–enabled. Unit 0 is assumed if no other unit number is given. If the specified device is hardware write–protected, a warning is issued.

ADEX requests a confirmation is the command is executed in manual mode. ADEX verifies the new status after the device is write–enabled.

WARNING

> It is necessary to verify that the media you are unprotecting does not contain any DGC or customer software. Once the media is write–enabled, the data on the media is vulnerable.

Example

ADEX-CLI> U *22*

Change status of device code 22, unit 0, to write–enabled.

WRITE

Syntax: **Write** *[message]*

Description

WRITE is used in macros to print a message, followed by a carriage return. It informs the operator of the occurrence of an event. A blank line is printed if no message is stated.

If a character sequence with the format *xx* is encountered in *message, xx* is treated as a number, and only the ASCII character with the value of the number specified is printed. Only the least significant 7 bits of the number is interpreted. *xx* is parsed according to the current radix.

Example

ADEX-CLI> W ***Testing Completed***

This message will be printed out.

XEQ

Syntax: **[XEQ]** *filename[.[prg,togl]] [passcount [start_adr]]*

Description

XEQ loads the specified file into memory starting at the physical location 0 and executes it. If a file type is not given, then a search is made, first through ADEX program files, then through toggle program files. If you indicate a file type, then the search takes place only on that type.

The pass count is the number of passes a program makes before returning to ADEX. If you do not specify a count, the default *PASMAX* is used (as defined by the DEFAULT PASMAX command). The initial default PASSMAX is 1.

If you do not give a starting address, program files are started at location 200 (8). Toggle files are started at location 0.

When executing a DTOS program, a pass count of −1 causes the program to run in a DTOS manual mode. Pass counts other than −1 cause the program to run in a DTOS automatic mode for the specified number of passes.

You can omit the **XEQ** portion of the execute command and simply type in the filename and any arguments.

Example

ADEX-CLI> **X** *6236_FUNC 3*

Loads the program file, *6236_FUNC 3*, into memory and executes it. It also changes the number of passes the program makes to three.

SECTION 3
ADEX UTILITIES

SECTION 3
ADEX UTILITIES

This chapter describes the most frequently used ADEX utilities. Other utilities, less frequently used, are found in Appendices F and G. Use the ADEX utilities to help you troubleshoot your system.

3.1 THE ADEX MEDIA BUILDER (AMB)

The ADEX Media Builder (AMB) is a utility for creating new ADEX media from the ADEX runtime media. There are two options: making an exact replica of the runtime media or selecting certain files from which to build.

The ADEX Media Builder can create a system on:

- Scratch Media
- ADEX Media
- CORESIDENT ADEX Disk

Check on the status of the media with the **EQUIPMENT** command. If using Scratch media which is write-protected, change its status with either the **UNPROTECT** or **SCRATCH** commands or with the Manual Sizer (MSIZE). When building or updating a CORESIDENT disk, ADEX ignores write-protection. Refer to 3.1.5 for more information on CORESIDENT. When updating a non-CORESIDENT ADEX disk, ensure that it is classified as media type 4 (ADEX).

The media Builder operates in two modes:

- Automatic
- Manual

By setting switches on the Switch Register, you are able to use Multiple Media Building, Customized Building, and CORESIDENT mode.

NOTE

The Media Builder automatically creates permanent files.

3.1.1 Automatic Mode

Use automatic mode to eliminate operator questions. Instead of taking all the defaults in manual mode, to save time, run in automatic mode. There is also the advantage of being able to execute the automatic mode of the media builder from a macro script.

In automatic mode, the Media Builder builds a system on the first scratch media (or CORESIDENT disk) it finds in the System Equipment Table (the table is ordered by device code and unit number) unless otherwise instructed by SWREG. The Builder copies the files from the runtime media to the build device. To control which device the Media Builder uses, make sure that all the devices ahead of it in the SET have write protection.

Before executing the Media Builder in automatic mode, set the Switch Register (SWREG). These are two 16-bit registers which allow information to be passed to programs. See Table 3-1 for a summary of all the bits in SWREG.) To run in automatic mode, set bit 15 to 1 and bits 16-31 to the model number of the device on which you are going to boot the media you are building.

To do this, use the SWREG command, followed by two arguments: a 1 and the model number. For example, to build on a 6236 disk drive, type:

ADEX-CLI> **SWREG** *1 6236.*

Notice the period (.) after 6236, indicating that the model number is in decimal, not octal.

To execute the Media Builder, type:

ADEX-CLI> **XEQ** *AMB*

or

ADEX-CLI> **AMB**

When the Media Builder finishes, it then sends a message saying it has built the media.

Table 3-1. SWREG Summary

BITS	DESCRIPTION
0-1	Unused.
2-3	0,0 = The build media has to be manually booted during power-up. The "FILENAME [ADEX]?" and "Run Autosizer?" questions will have to be answered by the operator. The remaining three options: (0,1), (1,0), and 1,1) will cause the build media to auto-boot during power-up. The FILENAME [ADEX]? question will not appear, but the default [ADEX]? will be chosen. The difference between these three choices involves the Autosizing options: 0,1 = The Autosizer will not run. 1,0 = The Autosizer will run, but the output from the sizer will not be printed to the screen (same as Reporting Level 0 during manual boot.
4	0 = Help files are not transferred to new build media. 1 = Help files are transferred to new build media.
5	0 = Runtime disk will not transfer the AMB files to the new build media. 1 = AMB files will be included in the new build.

Table 3-1. SWREG Summary (continued)

BITS	DESCRIPTION
6	0 = Build a normal (Advanced) Mode media. 1 = Build a CORESIDENT (Advanced) Mode media.
7	0 = Build a normal (Advanced) Mode media. 1 = Build a Customer Mode media.
8	0 = Normal build. 1 = Customized build using System Equipment Table (SET) or a TXT file specified in STRING.
9	0 = Customizes the build to the current equipment table if STRING is null; or uses a special equipment table in the UDF file specified in STRING. 1 = Customizes the build by including only those files listed in the TXT file specified in STRING.
10	0 = Do not optimize disk space utilization on multiple media builds (diskettes). 1 = Optimize disk space utilization on multiple media builds.
11	0 = Unknown disk media and tape media are write-protected. Unknown disk media and tape media are write-enabled or scratch. NOTE Unknown disk media refers to non-ADEX media.
12	0 = Installs a new system bootstrap and does a surface analysis on the media (initializes the media). Any files currently on the media are destroyed. All selected files are transferred to the media. 1 = Does not install a new system bootstrap. Uses the free file space currently defined on the media. Those files already on the media are not destroyed. Only lower revision files are replaced by newer ones and/or new files are appended to the media. Bit 12 is only applicable if Bit 15 = 1.
13	0 = Reports block number of unusable sector and builds around or restarts disk build. 1 = Aborts disk build upon finding an unrecoverable unusable sector.
14	0 = Does not log transferred files to the system console. 1 = Logs transferred files to the system console.
15	0 = Manual mode of operation (questions asked). 1 = Automatic mode of operation (no questions asked).
16-31	If bit 15 = 1, then bits 16-31 contain the boot media model number, entered in decimal notation followed by a period (.).

3.1.2 Manual Mode

To select files for the new media or to build several media in one sitting, use the Media Builder in manual mode. Refer to Figure 3-1 for a flowchart of media building in manual mode.

Before executing the program:

- Make sure that bit 15 of the switch register is set to 0, otherwise the builder will assume the automatic mode.

NOTE

SWREG is always cleared when a program returns, when an abort sequence is executed, and when ADEX is brought up.

- To have tracking information appear as the program is running, set bit 14 of the switch register to 1. By doing this, it is possible to see which files have been transferred.
- To take the second option, turn off the console logging with the LOG command to avoid wasting log space.
- To make the media builder abort if it encounters an unusable sector, set bit 13 of the switch register to 1. If this bit is set and the Builder finds an unusable sector, you get an error message. If you do not set bit 13 and the Media Builder finds an unusable sector, the program builds around the unusable sector after it informs you that it has found one.

Execute the Media Builder in the same way as in automatic mode:

ADEX-CLI> **[XEQ]** *AMB*

The program begins by asking four questions:

1. `BOOT MEDIA MODEL NUMBER?`

   Enter the model number of the device on which you are going to boot the media you are building (for example: 6236). The program repeats this question if the model number given is not supported or if the System Equipment Table does not contain a device on which you can build media with that model number.

2. `DEVICE CODE [xx]?`

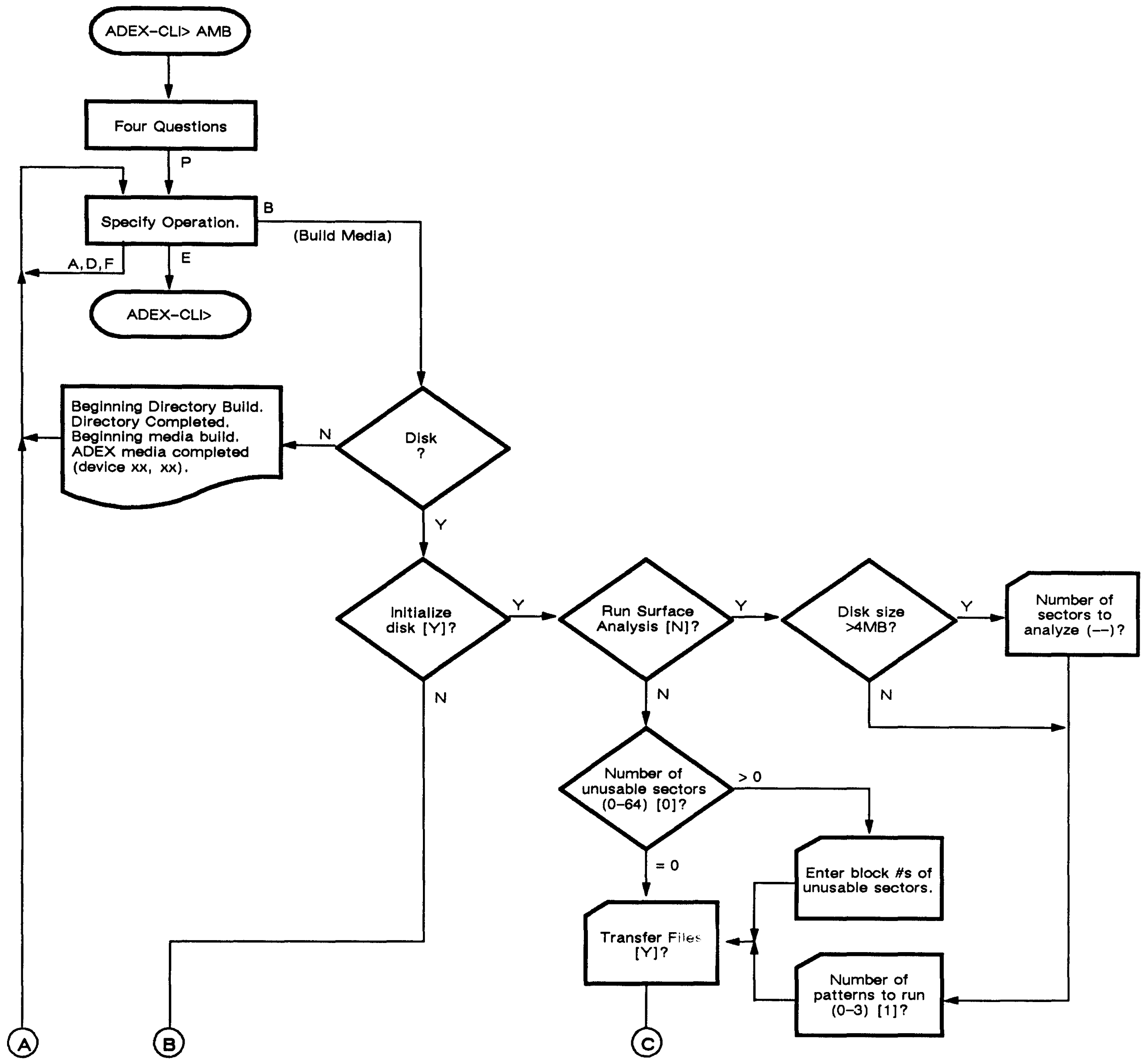
   Enter the device code of the drive on which the media is being built. The default is the first scratch media that the Media Builder finds in the System Equipment Table that is capable of building a media of the model number Specified. (Press NEW LINE if you want the default.)

3. `UNIT NUMBER [xx]?`

   Enter the unit number of the drive on which the media is being built. The default is the first scratch media found in the system Equipment Table that is capable of building a media of the model number specified.

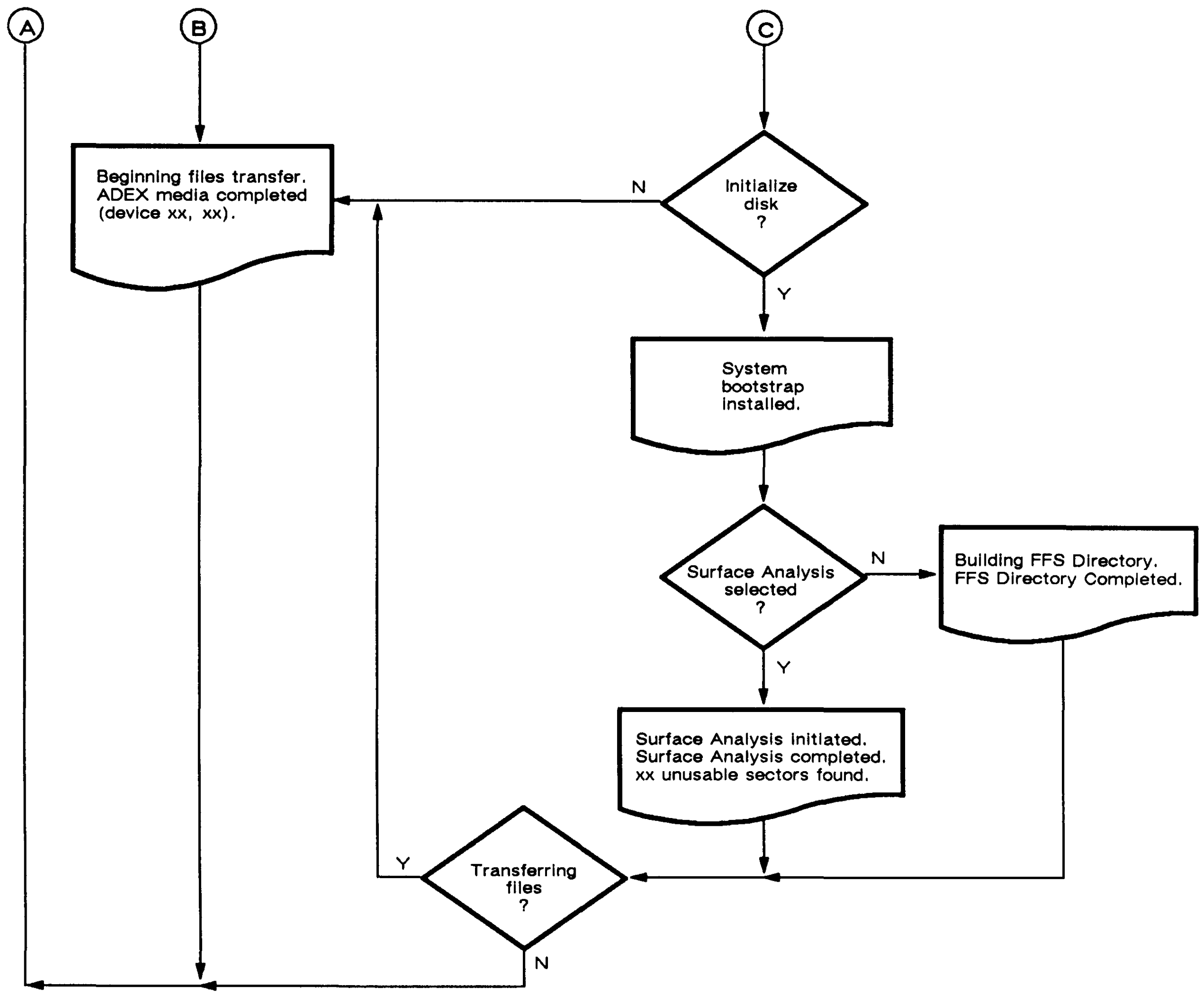
4. `SYSTEM I.D. [xxxxxxxx]?`

   Enter the system identification message for the media being built. The default is the runtime media's identification message.



FS-12832

Figure 3–1. Manual Mode Flowchart (Sheet 1 of 2)


A
B
C
Beginning files transfer.
ADEX media completed
(device xx, xx).
N
Initialize disk ?
Y
System bootstrap installed.
Surface Analysis selected ?
N
Building FFS Directory.
FFS Directory Completed.
Y
Surface Analysis initiated.
Surface Analysis completed.
xx unusable sectors found.
Y
Transferring files ?
N

FS-12832

Figure 3-1. Manual Mode Flowchart (Sheet 2 of 2)

The Media Builder now lists a choice of operation:

```
        SPECIFY OPERATION (?,A,B,D,E,F,P) [B]?
```

The default is to build the media. Table 3-2 summerizes the commands.

Table 3-2. AMB Commands in Manual Mode

COMMAND	MEANING
?	Briefly explains the options.
A	Adds files to the list of enabled files which tell the Media Builder which files to copy from the runtime media to the new media.
B	Builds the media.
D	Deletes files from the list that tells the Media Builder which files to copy from the runtime media to the new media.
E	Exits from the Media Builder and returns to the CLI.
F	Indicates which files are enabled for Media Build.
P	Redefines the parameters for the new media (i.e., asks the initial four questions again).

3.1.2.1 The ? Command – Use the ? command for a brief explanation of the choices available.

3.1.2.2 The A, D, and F Commands – In manual mode, there is the option of copying selected files by manipulating the selection list. Initially, this list includes all the files on the runtime media (unless the Custom Building option is being used). However, the **D** command may be used to delete files from the list.

When using the **A**, **D**, or **F** command, it is possible to refer to the files directly by name or use the * and + templates as short-cuts. The * matches single characters; the + matches any number of characters. Templates that do not contain a # prefix refer only to user files. Those with a # prefix refer to system files. (Note that the # is not parsed as part of the filename.)

The system prompts you for specific file information with the message:

```
        SPECIFY FILENAMES/TEMPLATES?
```

You can answer this question with one or more filenames and/or templates, separated by commas.

Both system files and user files can be deleted or inhibited. Even though system files are required by ADEX in order to run, some are utilities that may not be needed on the media that is being built. If templates are used, be sure not to inadvertently delete any system file that is needed. The **D** command confirms action by stating that it has inhibited a file.

NOTE

If a system file is inhibited, the AMB will ask for confirmation before allowing the media to be built.

Files are also deleted by families. A family consists of a parent file, such as a program, plus any other subordinate files that the parent requires, such as overlay files. Refer only to parent files. Their families are included automatically. Some subordinate files are shared by several parent files. If you delete a family which contains a shared subordinate, this file is not deleted if it is still required by another family. To see which files are parents and which are subordinates, use the **F** command.

Use the **F** command to find out the status of a file. By using the **F** command and a + argument, it is possible to view all the files (templates are allowed). The **F** command first lists the files that are enabled and then those which are inhibited. It further distinguishes between sytem and user files. System files are preceded with *. Also subordinate files are indented to indicate that they are part of a family. For example:

```
PROG1.PRG
   ULIB0.OLF
   ULIB2.OLF
*ACOMP.PRG
PROG3.PRG
   ULIB1.OLF
   ULIB2.OLF
```

means that ULIB0 and ULIB2 are overlay files that are part of the PROG1 family. ULB2 is also part of the PROG3 family, making it a shared subordinate. The asterisk (*) before ACOMP.PRG identifies it as a system files.

If files are inadvertently deleted, use the **A** command to add them back to the selection list. Either the filenames or the templates can be used when referring to files. The **A** command confirms its actions by stating that it has enabled a file.

3.1.2.3 The B Command – Once you have selected files to copy, build the media with the **B** command.

NOTE

> To exit the build operation when building to a disk, press the ESC key in response to any question (a $ will appear on the screen) and you will return to the SPECIFY OPERATION question. No action is taken in the operation you initially chose.

If any system files are inhibited, a confirmation is required before building the media:

```
SYSTEM FILE OR FILES INHIBITED. CONFIRM (Y,N) [N]?
```

To build the media, answer Y; otherwise you will return to the Selective Builder prompt.

When building a media on tape, the program issues messages informing you of its progress:

```
BEGINNING DIRECTORY BUILD.
DIRECTORY COMPLETED, BEGINNING MEDIA BUILD.
ADEX MEDIA COMPLETED (xx,xx).
```

When building on a disk, you go through several question and answer steps. The sequence of questions and messages is shown in Figure 3-1.

1. The AMB begins by asking:

   ```
   INITIALIZE DISK [Y]?
   ```

   If you answer Y, the program destroys any data that was previously on the disk and installs a new system bootstrap. You then have the choice of running a surface analysis on the disk.

2. If you initialize the disk, the AMB asks:

   ```
   RUN SURFACE ANALYSIS [N]?
   ```

   If you choose [N] to Run Surface Analysis question, you must answer the questions in Step 4. If you do choose to run it, go to step 3. You should answer No to the transfer files question only if you are using the Builder to initialize disks.

3. If you are analyzing the surface, and the capacity of the disk is greater than 4 MB, the AMB asks:

   ```
   NUMBER OF SECTORS TO ANALYZE (MIN= 1000,MAX= xxxxxx) [1000]?
   ```

   You must supply the number of sectors to analyze if the disk's capacity is greater than 512 KB. You must also choose the number of surface analysis patterns to run.

   Analyze more than 8192 sectors only if you expect to use more than 8192 sectors of the build media. Regardless of disk size, the builder then asks:

   ```
   NUMBER OF PATTERNS TO RUN (0-3)[1]?
   ```

   If you choose 0 patterns, the analysis is based on the hardware bad sector flag of each sector, if it exists. If you answered Y to the initialize disk question, then the transfer file question is asked here.

   If you selected initialization, surface analysis, and files transfer, the output would appear as follows:

   ```
   SYSTEM BOOTSTRAP INSTALLED.

   SURFACE ANALYSIS INITIATED.
      SURFACE ANALYSIS COMPLETED, xx UNUSABLE SECTORS FOUND.

   BEGINNING FILES TRANSFER.
   ADEX MEDIA COMPLETED (DEVICE xx,xx).
   ```

   To see how many sectors still have to be analyzed during the surface analysis process, press the ESC key. The AMB provides the answer in the form of a 32-bit decimal number.

   If you did not select file logging by setting SWREG bit 14, hitting ESC when the AMB is transferring will cause the AMB to print the file index number and name of the last file transferred to the build media. Also, CTRL-C-ESC causes the remainder of the files NOT to transfer. The message FILE TRANSFER ABORTED is printed.

NOTE

System response will not be immediate.

4. If you chose not to analyze the surface, the AMB asks the following:

```
NUMBER OF UNUSABLE SECTORS (0-64) [0]?
   (1)? xxxxxx
   (2)? xxxxxx :     :
   (xx)? xxxxxx
```

   The last four lines indicate that when you enter the number of unusable sectors, the AMB then asks you the block numbers of the unusable sectors. (This information is available if you had previously initialized and/or built to the build media.) If you do not know any of the unusable sectors, use the default answer of 0.

   If you selected initialization without surface analysis, the surface analysis message would be replaced with:

```
BUILDING FREE FILE SPACE DIRECTORY.
      FREE FILE SPACE DIRECTORY COMPLETED.
```

   If you requested tracking by setting SWREG bit 14 , you will also see the index numbers and names of all the files that are being copied to the new media.

3.1.2.4 The E Command – Use the **E** command to exit from the Media Builder and return to the CLI.

3.1.2.5 The P Command – The **P** command is very useful when building several media at one time. The command results in the AMB asking the four questions about model number, device number, unit number, and system ID. The defaults are the same as those of the last media built. You can change some or all of the parameters.

3.1.3 Multiple Media Building

When the space required to hold all of the selected files exceeds the capacity of the build media, the AMB will continue its build on another media if the media is a removable disk. This process is called multiple media building. During multiple media building, system files are repeated on every media. For user files, the AMB takes care not to break up families when deciding which files go on which media. For example, if a program exists on a media built by the AMB, it is guaranteed that all of its subordinate files are also on the media.

3.1.4 Customized Building

The AMB can build a disk with a customized set of files. The list of files transferred can be customized via the equipment table or a user-specified files list.

When you customize via the equipment table, it selects the system files and all ADEX programs that test equipment listed in either the current System Equipment Table or an equipment table saved in a UDF file. To customize via the equipment table, set SWREG bit 8 and clear SWREG bit 9. If customizing to the current SET, then clear the STRING buffer also. To customize to an equipment table saved in a UDF file, enter the name of the file in the STRING buffer. The build is aborted if the file specified does not exist.

When customizing to a user-specified files list, only those files that are listed in specified TXT file are selected. To customize via a user-specified files list, set SWREG bits 8 and 9. Then, enter the name of the TXT file that contains the files list in the STRING buffer. The builder ignores any files in the user-specified Files list that do not exist on the runtime media. Figure 3-2 gives an example of a custom files list.

```
CCMSZ.RYSY
MHMINI.RYSY
KERNEL.RYSY
SINIT.RSYS
ASIZE.PRG
MSIZE.PRG
CSMOD1.OLF
CSMOD2.OLF
CSMOD3.OLF
SPMON.OLF
ERMSG0.SDF
ERMSG1.SDF
ERMSG2.SDF
ERMSG3.SDF
ERMSG4.SDF
ABORT.SNRU
BYE.SNRU
CHAIN.SNRU
CLI.SNRU
CLIERR.SNRU
CLISR.SNRU
ECLASS.SNRU
EQUIPMENT.SNRU
ERRMSG.SNRU
EXCCHR.SNRU
FINDEV.SNRU
GETDRV.SNRU
ONEK.SNRU
SEDIT.SNRU
SWEG.SNRU
ENDB.UDF.O
ARI.SRCH.O
CSP.SRCH.O
DCU.SRCH.O
DPP.SRCH.O
IAC.SRCH.O
IPP.SRCH.O
ISP.SRCH.O
LMP.SRCH.O
LPP.SRCH.O
MCP.SRCH.O
PTP.SRCH.O
SCM.SRCH.O
TTS.SRCH.O
ZBP.SRCH.O
GENMN.OLF.O
GMON.OLF.O
CLI.UDF.O
```

FS-12833

Figure 3-2. Custom Files List

3.1.5 CORESIDENT Mode

A CORESIDENT build allows the ADEX diagnostic operating system to co–exist on the same disk as your operating system (i.e., AOS, AOS/VS, DG/UX, RDOS). The two systems can not execute at the same time. Entry to the ADEX operating system is through your operating system's boot program. In response to the boot program's prompt, enter the name of the ADEX Link program. For AOS and AOS/VS, this is "ADESL". For RDOS systems, enter "ADESL.[SV]" (.SV is optional).

NOTE

The CORESIDENT feature is not available on NOVA or microNOVA systems as disk capacities are not large enough.

3.1.5.1 REQUIREMENTS

1. Your software must support the CORESIDENT feature. See Table 3–3. You must have specified a diagnostic area when your disk was initialized, using DFMTR for AOS or AOS/VS, DKINIT for RDOS, or BADBLOCK for DG/UX. Consult the appropriate Release Notice which accompanied the software.

Table 3–3. System Software and Disk Utilities

OPERATING SYSTEM	REVISION	DISK FORMATTING UTILITY
AOS/VS	REV 6.04 or higher	DFMTR
AOS	REV 7.03 or higher	DFMTR
RDOS	REV 7.50 or higher	DKINIT
DG/US	REV 3.00 or higher	BADBLOCK
DG/RDOS for MV/2000 DC	REV 2.00 or higher	DKINIT
AOS/DVS	REV 0.00 or higher	DFMTR

2. You must allocate the diagnostic area correctly when initializing the disk. Refer to the "CORESIDENT DIAGNOSTIC SYSTEM OPERATOR'S REFERENCE GUIDE" (DGC PART No. 014–001186).

3. Your ADEX media must support the CORESIDENT feature. See Table 3–4. Consult the Release Notice.

Table 3–4. ADEX Media with CORESIDENT

MODEL NUMBER	DESCRIPTION	LINK PROGRAM (#) (3 FORMATS AVAILABLE)
30200–20H	MV ADEX REV 4.0	AOS/VS (1), RDOS (2), DG/UX (3)
30200–20C	MV ADEX REV 4.0	AOS/VS (1), RDOS (2), DG/UX (3)
30200–20G	MV ADEX REV 4.0	Last 3 Diskettes (as listed above)
30199–20M	N/E ADEX REV 5.0	AOS (1), RDOS (2)
30199–20H	N/E ADEX REV 5.0	AOS (1), RDOS (2)
30199–20C	N/E ADEX REV 5.0	AOS (1), RDOS (2)

H = 1600 BPI, C = Cartridge Tape, M = 800 BPI, G = 96 TPI Diskette

3.1.5.2 Initial CORESIDENT Build – The following procedure outlines the steps involved when doing the initial CORESIDENT build. You must complete one step before proceding to the next step.

1. Make sure you have backed–up the system using the (Logical) DUMP command.

CAUTION

PCOPY or BURST must not be used, since the physical format of the disk changes!

2. Make sure that the diagnostic area has been correctly allocated. Refer to the "CORESIDENT DIAGNOSTIC SYSTEM OPERATOR'S REFERENCE GUIDE" (DGC PART No. 014–001186).

3. Load the ADEX Link program into the root directory. This is a stand–alone program on your ADEX tape that provides a link between the operating system and the diagnostic area on the disk. It is executed via the system boot program. The link program is found on File 1 on MV ADEX and NE ADEX tapes for AOS and AOS/VS systems, File 2 for RDOS systems, and File 3 for DG/UX systems.

   Example:

   ```
   "LOAD/V @MTB0:1"
   ```

4. Ensure that you bring down the operating system and that all devices are ready and on–line.

5. For AOS and AOS/VS systems, if you are using a multiple (physical) disk Logical Disk Unit (LDU), record the model number, device code, and unit number of the last physical disk in the logical disk.

6. Build CORESIDENT ADEX on the system disk, ensuring that the System Equipment Table is accurate and SWREG Bit 6 is set. For Auto Build, set SWREG to 1203,6XXX. Note that 6XXX is the disk model number in decimal, noted by (.).

NOTE

For more information on CORESIDENT mode, refer to Appendix E.

3.1.6 Updating AMB Built Media

Standard or CORESIDENT ADEX disk media can be updated using subsequent releases of ADEX media, as long as the build media is a higher (major) revision than that which was previously built on the disk. For example, a disk built with MV/ADEX Rev 5.0 can be updated using MV/ADEX rev 6.0, by the additional setting of bit 12 in the SWREG. Only those files that are higher revisions of the same filename will be transfered, thus replacing (deleting) the lower revision of the files. Also note, any newly required files not on the disk media will be appended in the free file space.

3.2 THE EDIT UTILITY

The Edit Utility lets you edit any type of existing file and create source, text, or toggle files. Edit not only serves as a text handling utility, but also a patch utility for putting fixes into programs. If editing a nontext file, the data in the file appears in numerical form (in the current radix). When working with source or toggle files, you can execute them right from the Edit utility.

NOTE

You cannot execute Edit within a script.

The Edit utility numbers the lines in a file for your convenience (the numbers are not part of the file). If working with ASCII text, the line numbers are decimal. Otherwise, they are in the current radix. Most Edit commands work with one entire line or a group (range) of lines. As a file is modified, the line numbers are automatically updated to reflect additions or deletions.

When editing a file, you move the cursor from line to line. The current line is the one on which the cursor is currently sitting. This is the line that you last displayed, modified, inserted, or appended. When you start editing a file, the current line is the first line of the file.

Control characters that are either entered on the command line or exist in the text file being edited are displayed on uppercase video display terminals as blinking characters and are displayed on upper/lowercase video display terminals, graphic displays and color displays as dimmed characters. However, the **LIST** command prints the control characters literally (e.g., a <14> will cause an ERASE PAGE character to be printed instead of a blinking or dimmed "L").

If working on a terminal that supports dimmed characters but prefer to see the control characters blinked instead of dimmed, you can resize (via the Manual Sizer) the system console as an uppercase video display terminal. If working on an uppercase video display terminal and do not want the control characters blinked at all, type CTRL-P CTRL-D to disable blinking (CTRL-P CTRL-C re-enables blinking).

3.2.1 Executing Edit

Before executing Edit, make sure the OPERATOR flag is on. Then type:

ADEX-CLI> **EDit** *filename.filetype*

where *filename.filetype* is the name and extension of the file that you want to create or edit. If the filename and/or extension is omitted, Edit prompts you for it. If the file exists, you get the Edit prompt:

```
EDIT-CLI>
```

If you indicated a source (SRCE), text (TXT), or toggle (TOGL) file and the file does not exist, the Edit asks:

```
CREATE NEW FILE (Y,N) [N]?
```

If you answer Y, you will get the EDIT-CLI prompt. Otherwise you get the ADEX-CLI prompt. Once you have the Edit prompt, you can enter any Edit command.

3.2.2 The Edit Commands

Table 3–5 summarizes all the Edit commands. Following the table, the commands are discussed in detail.

Table 3–5. Summary of Edit Commands

COMMAND	MEANING
ABORT	Discards all changes and returns to ADEX–CLI.
APPEND	Appends lines to file.
BYE	Saves the file on runtime media and returns to ADEX–CLI.
DELETE	Deletes one or more lines.
EXECUTE	Executes the source or toggle file.
INSERT	Inserts text into the file.
LIST	Displays one or more lines of the file.
MODIFY	Modifies one or more lines of the file.
SAVE	Saves the file on the runtime media and returns to ADEX–CLI.
STATUS	Displays information about the file being worked on.

ABORT

Syntax: **ABort**

Edit then asks:

```
          DO YOU REALLY WANT TO DISCARD YOUR EDITS (Y,N) [N]?
```

Answer Y or N. The default is N.

ABORT discards any changes that were made and returns you to ADEX CLI. If the file you were working on was created during the Edit session, the file is deleted. When you use this command, Edit asks you to confirm your action. If you do not confirm, you get another Edit prompt.

APPEND

Syntax: **APpend**

APPEND adds (appends) lines to the end of the file you are editing. To add lines in the middle of the file, use to INSERT command. After you type in the **APPEND** command, an asterisk (*) appears before the line number to indicate that you are in the append mode. Type in the lines you want to add, concluding each line by pressing NEW LINE. When finished, press ESC to get back to the EDIT–CLI prompt.

BYE

Syntax: **Bye** *[filename]*

BYE saves the file being worked on and returns you to ADEX CLI. This command is identical to Edit's **SAVE** command.

DELETE

```
Syntax: Delete   [ALL]
                 [LAST]
                 [line_number_ [line_number]
                               [LAST]
```

ALL means the entire file. *LAST* means the last line of the file. You can provide two line numbers to indicate a range (that is, a block of lines) or a line number and the word *LAST* to indicate a block from some line in the file to the last line.

NOTE

Line numbers must be in the current radix. You may use a radix indicator.

DELETE gives you the choice of deleting the current line, all the lines, the last line of the file, or a range of lines in the file. For example:

EDIT–CLI> **D**	Deletes the current line.
EDIT -CLI> **DELETE** *ALL*	Deletes the entire file.
EDIT–CLI> **DEL** *LAST*	Deletes the last line of the file.
EDIT–CLI> **D** *3*	Deletes the third line.
EDIT–CLI> **D** *3 7*	Deletes lines 3 through 7.
EDIT–CLI> **DELETE** *21 LAST*	Deletes from line 21 to the end of the file.

EXECUTE

Syntax: **Execute** *[togl_start_address]*

togl_start_address is the starting address for the toggle file. (If an address for a source file is provided, you will get an error message.) If you try to execute a file other than a source or toggle file, you get an ILLEGAL FILE TYPE message and an Edit prompt. If the file is empty, you get an EMPTY FILE message and an Edit prompt. If everything is correct, you are asked to confirm the execution:

```
        CONFIRM (Y,N) [N]?
```

Y results in execution. N gets another Edit prompt. You cannot return to ADEX from the toggle program execution. You must manually reboot the media. When the script completes, you are placed back in ADEX-CLI.

This command executes a toggle file or a one page (256 word) source file. If the source file is longer, you will get the message MEMORY RESOURCES NOT AVAILABLE. Also, this macro file may not call other macros. If the edit file is a toggle file, the program you execute is moved to location 0. Execution starts at location 0 unless you provide a starting address.

NOTE

Any changes made during the Edit session are not saved. To save the edits, use the **SAVE** (or **BYE**) command.

INSERT

Syntax: **Insert** *[line _number]*
[LAST]

Without an argument, **INSERT** inserts text before the current line. Otherwise, you can indicate a specific line with *line_number* or the last line with *LAST*. For example:

```
EDIT-CLI>I 20
*20  This text is inserted before line 20.
*21 $

EDIT-CLI>I LAST
*xxx  This text is inserted before last line in file.
*xxx $
```

INSERT lets you add up to 256 words (512 characters) in one operation. On a 32 KW (or greater) machine, you can have files up to 21.5 KW. On a 16 KW word machine, your files can reach 5.5 KW. If you exceed this limit, you get an error message saying INSERT BUFFER FULL.

When you use **INSERT**, you enter insert mode, indicated by a * prompt. This is followed by a line number (for text files) or a word address (for other files). Type in the data you wish to insert. To exit from the insert mode, use ESC for text files and any non-numeric character, NEW LINE, or Carriage Return for other files.

LIST

Syntax: **List** *[ALL]*
[LAST]
[line_number [line_number]]
[LAST]

ALL means the entire file; *LAST* means the last line of the file; and *line_number* means a particular line, referred to by line number. Line numbers apply only to text files. For other files, you must provide the relative word address. You can provide two line numbers to indicate a range (a block of lines) or a line number and the word LAST to indicate a block of lines from a point in the file to the end of the file.

The **LIST** command lets you display the current line, the entire file, the last line of a file, or a range of lines. For example:

EDIT–CLI> **L**	Displays the current line.
EDIT–CLI> **LIST** *ALL*	Displays the entire file.
EDIT–CLI> **LIST** *LAST*	Displays the last line of the file.
EDIT–CLI> **L** *3*	Displays the third line.
EDIT–CLI> **L** *3 7*	Displays lines 3 through 7.
EDIT–CLI> **L** *21 LAST*	Displays from line 21 to the end of the file.

MODIFY

```
Syntax: Modify  [ALL]
                [LAST]
                [adr [adr]]
                     [LAST]
```

ALL is the entire file; *LAST* is the last word of the file; *adr* is the specified address (or range of addresses); and *adr LAST* is the range of addresses between adr and the last word of the file.

MODIFY allows you to modify lines in a source file, but only when the system console is a video device. You can modify the lines using the Screen Edit Utility. The prompt is the line number followed by the source file line. You can exit Modify prior to the last line specified by typing ESC.

You can modify lines in numeric files regardless of console type. You will get a prompt that is the word address, followed by the current value at that address. Then, type in the new numeric data, followed by NEW LINE. If you typed in a non-numeric character while entering numeric data, you exit out of the Modify mode and return to EDIT-CLI.

SAVE

Syntax: **SAve** *[filename]*

SAVE saves the file on the runtime write-enabled media. It returns to ADEX-CLI.

If a filename is not provided, or if a filename is provided and it is the same as the filename specified when the Edit utility was invoked, then the filename provided with the Edit command is used.

If a filename is provided on the **SAVE** command that differs from the one specified on the **EDIT** command, then if a file was created when the Edit utility was invoked, it is renamed to the filename specified on the **SAVE** command. If the file specified on the **EDIT** command already existed, then specifying a new filename on the **SAVE** command gives you the option of deleting the old file.

The **SAVE** command is ignored if there isn't enough space for the edited file on the runtime media and a warning is issued: FILE SPACE EXHAUSTED. If the file is empty, EMPTY FILE appears, and a new EDIT-CLI prompt is issued.

NOTE

You cannot use **SAVE** if the runtime media is a tape. Use **EXECUTE** or **ABORT** to exit from EDIT when running off tape.

STATUS

Syntax: **STatus**

STATUS gives the following information on the file: name, type, current line number or word address, last line number or word address, number of 256–word pages in Edit file, whether the Edit file is empty, and/or execute only, if applicable.

For example:

EDIT–CLI> **ST** results in the following:

```
FILENAME: xxxxxxxxx         FILE TYPE: xxxxxxxxxx
CURRENT LINE: xxxx          LAST LINE: xxxxx

NUMBER OF PAGES: xx
EMPTY FILE:
EXECUTE ONLY FILE:
```

The last two messages are printed only when they apply.

3.3 THE OCTAL DEBUGGING TOOL (ODT)

The Octal Debugging Tool (ODB) provides some limited debugging facilities. The advantages of using the ODT rather than the Symbolic Debugger (ADEB) is that the ODT does not use any ADEX system routines. Therefore, it does not have to rely on the ADEX software in order to operate. The ODT, however can only be executed from a primary teletype (device code 10/11), an MBC (device code 20), or an MPT memory mapped screen. In an MPT, ADEX system routines are used. Breakpoints defined with the ODT are automatically removed once the breakpoint is entered.

The ODT can be used for system restarts (if the ODT can be accessed at location 202) and memory dumps, as explained under the M command.

The prompt sign for the ODT is the at-sign (@). This prompt signals that the ODT is ready for you to input commands. If you type in an illegal command or expression, the ODT prints a question mark and gives a new prompt.

3.3.1 Access to ODT

The ODT can be accessed in four ways:

- By using breakpoints.
- By issuing a system call command (type CTRL-O).
- By jumping to location 202 or starting a hard/soft console at location 202.
- By a system panic.

NOTE

Access into ODT from CTRL-O applies only when running ADEX-CLI. When running MV-ADEX user programs, i.e. E80CPUV2, CTRL-O will halt machine and return you to SCP-CLI>.

If running .DTOS programs, it will enter ODT.

When breakpoints are used, the **P** command will continue execution at the breakpoint address. The system call command format is CTRL-O [SCALL ODT]. If the ODT is accessed in this manner, the **P** command returns to call +1. After accessing ODT at location 202, the **P** command does a system restart. A restart involves clearing key system variables, executing an abort (CTRL-C CTRL-B) sequence and issuing a new ADEX-CLI prompt. If the ODT is accessed through a system panic, the **P** command will do a system restart.

3.3.2 Command Formats

ODT commands consists of either a letter or an expression followed by a letter. An expression is one or more numbers separated by a plus (+) or minus (–) sign. The period symbol (.) can be used in place of a number anywhere in an expression. The value of the period symbol is equal to the total value of the last expression.

3.3.3 Commands

When using the ODT commands, you must provide an expression or a number to open either a memory location or an accumulator, respectively. For example, typing in the expression 1000 + 23 would open a particular memory location. When a memory location or accumulator is opened, its contents are printed out. After the contents are printed, you can be alter them by typing a new number. When the location is closed, the typed-in number is stored in the previously opened memory location or accumulator. To open an accumulator, provide a number from 0–14, followed by the letter A. Each number represents a specific accumulator, as indicated in Table 3-6.

WARNING

Do not open any accumulators other than the ones in Table 3-6.

Table 3-6. Accumulator Numbers

NUMBER	ACCUMULATOR
0	AC0
1	AC1
2	AC2
3	AC3
4	Carry
5	ION Flag 0 = interrupts were off -1 = interrupts were on
6	Subtotal of current expression
7	Entry Flag -1 = CTRL-0 0 = Breakpoint
10	Dump media device code
11	Dump media unit number
12	Contents of location indicated by software frame pointer
13	Total of last expression (.)
14	Proceed address – do not modify this location

The remainder of the ODT commands are used to open and close memory locations and accumulators, to insert breakpoint instructions, to delete breakpoints, to start execution of a program at a certain location, and to perform memory dumps. Table 3-7 summarizes the rest of the ODT commands.

NOTE

The ODT accepts either uppercase or lowercase letters for alphabetic commands.

Table 3-7. ODT Commands

COMMAND	FUNCTION
expression/	Opens the memory location specified in expression.
NEW LINE (key)	Closes the currently opened memory location or accumulator (key) and a new prompt is issued.
Carriage Return (key)	Closes the current opened memory location or accumulator and opens the next location or accumulator.
Up Arrow (key)	Closes the currently opened memory location or accumulator and opens the previous location or accumulator.
/	Closes the currently opened memory location or accumulator and opens the memory location that is specified as the contents of the location or accumulator most recently closed.
expression B	Places a breakpoint instruction at location specified. When the specified instruction is executed, the ODT is re-entered and the actual instruction is restored.
B	Lists the location of the breakpoint defined by expression **B** command.
D	Deletes the breakpoint defined by the expression **B** command (in other words, restores the actual instruction at the breakpoint address).
P	Procedes according to the way the ODT was entered.
expression R	Starts execution at nonzero expression specified.
I	Performs an IORST instruction. CAUTION DO NOT execute this instruction if the system console or the secondary output device is on a multiplexor.

Table 3-7. ODT Commands (continued)

COMMAND	FUNCTION
expression =	Prints the value of the spcified expression and places in accumulator 13.
M	Dumps all logical memory to magtape on specefied device code and unit number, which are always 10A and 11A, respectively. You must mount a scratch tape and confirm the memory dump. You will return to the ODT when you remount the original tape and press any key. CAUTION Do not perform memory dumps while console logging is enabled. If you use the **P** command after a dump, this will result in a system restart. DO NOT use NEW LINE or a Carriage Return as a command terminator.

3.4 THE SCRIPT BUILDER (SCRBLD)

The ADEX Script Builder is used to generate a list of programs that run on the current CPU, that test equipment listed in the SET, and that can be executed without asking any questions. On a tape, these programs are in the form of a memory-resident script that is executed and lost when the script finishes. On a disk, however, there is a choice. The programs can be in the form of a memory-resident script or in a source file. The source file contains a series of execute commands which can be saved on the run time disk for later use.

Both files contain commands to test one or more parts of a system with a series of programs found on the runtime media. The script builder has three sources of information:

1. A cross-reference table that indicates which programs test which part of the system and on which CPU they run. This table is supplied by Data General. You should NEVER modify it unless instructed to do so.
2. The System Equipment Table.
3. A directory structure that lists which programs are on the media. This directory structure is built into the media when it is created. The structure is affected when you delete, add, rename, or update programs.

You can execute the Script Builder by typing in SCRBLD while in ADEX-CLI. SCRBLD is also executed automatically from RUN and ACCEPT macros. To run the Script Builder automatically, first turn off the OPERATOR flag with the **OPERATOR** command. Then type:

ADEX-CLI> **XEQ** *SCRBLD*

or

ADEX-CLI> **SCRBLD**

A one-page script is built and immediately executed. It is based on the System Equipment Table that is currently in memory. The program then gives the following messages:

```
BUILDING SCRIPT
SCRIPT BUILT
```

Once the script is executed, you return to the CLI prompt.

To run the Script Builder interactively, make sure the OPERATOR flag is on. Once the Script Builder is entered, you will be given a series of questions to answer, if the OPERATOR flag is set.

To execute the program, type:

ADEX-CLI> **XEQ** *SCRBLD*

or

ADEX-CLI> **SCRBLD**

You are now asked some questions:

NOTE

If the runtime media is write-protected, only the question in step 3 is asked.

1. The Script Builder asks:

   ```
   EXECUTE OR SAVE (E,S)?
   ```

   Answer **E** to execute and discard a script. Answer **S** to create and save a script cource file. If E, go to step 3. If S, go to Step 2.

2. If you answered **S** to the previous question, the Script Builder asks:

   ```
   SOURCE FILE NAME?
   ```

   Enter the file name you want for the source file. You can use the name of an existing source file. The Script Builder will update it.

3. Finally, the utility asks:

   ```
   EQUIPMENT TABLE [CURRENT]?
   ```

   Press NEW LINE to use the System Equipment Table currently in memory. Otherwise, enter the name of the file whose contents will be replaced with the SET. Figure 3-3 shows a sample interaction.

As shown in Figure 3–3, after answering the last question, the utility gives one of the following series of messages:

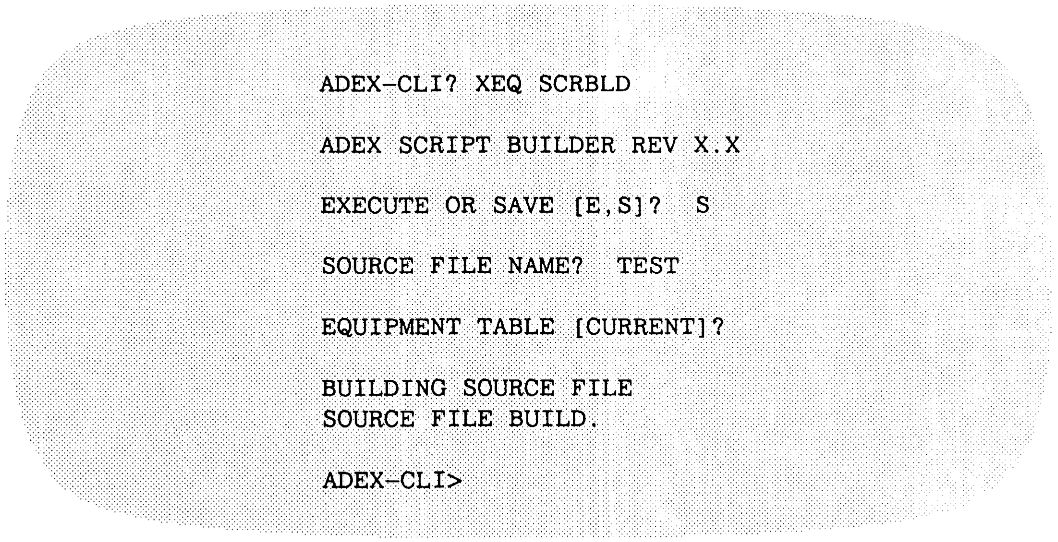
If you **EXECUTE**:

```
BUILDING SCRIPT
FILE SCRIPT FILE BUILT
```

If you **SAVE**:

```
BUILDING SOURCE FILE
SOURCE FILE BUILT.
```

You then return to the CLI prompt.



```
ADEX-CLI? XEQ SCRBLD

ADEX SCRIPT BUILDER REV X.X

EXECUTE OR SAVE [E,S]?  S

SOURCE FILE NAME?  TEST

EQUIPMENT TABLE [CURRENT]?

BUILDING SOURCE FILE
SOURCE FILE BUILD.

ADEX-CLI>
```

FS-12834

Figure 3–3. Sample Script Building Session

If you set the VERIFY flag in the Environment Control Word to YES, you can run only best-bet programs selected by the Script Builder.

3.5 THE UPDATE UTILITY

The Update Utility lets you update an ADEX system disk without rebuilding it. Currently, since AMB now allows updating of a ADEX system disk, Data General only uses 96 TPI diskettes as a source of new (updated) files. The source of the update is referred to as source media and is used to update disk runtime media. The source is only diskette, but the runtime media may only be disk.

3.5.1 Building Update Media

Although only new revisions of ADEX media are distributed by Data General, you can create your own update media by using AOS COPY commands. When you use AOS COPY commands, the first file on the tape must be the update directory, which defines the new System I.D. for runtime media , and lists files on the update tape. The first line in the Update Directory contains the new System I.D.; the second line describes File 1; the third line describes File 2; and so on. The format of these lines is as follows:

filename.filetype major_rev.minor_rev

where the major/minor revision is in decimal.

Example:

```
NEADEX_REV_00.01
MYFILE.PRG 3.0
MYSNRU.SNRU 1.0
```

The major/minor revision is in decimal.

When you copy SRCH files to the update tape, a record size of 4096 bytes must be used.

When using AOS, the new System I.D. must not exceed 79 characters and must be terminated by a NEW LINE character (ASCII <12>).

3.5.2 Executing Update Utility

Before executing the Update Utility, make sure that the source media you are using for the update is properly classified in the System Equipment Table. Using the Manual Sizer (MSIZE), add or modify the appropriate entry so that the classification is UPDATE (media type "6") for the source media (diskette). See Figure 3-4.

Because Update is interactive, make sure the OPERATOR flag is on before executing the utility. Execute the Update utility as follows:

ADEX-CLI> **XEQ** *UPDATE*

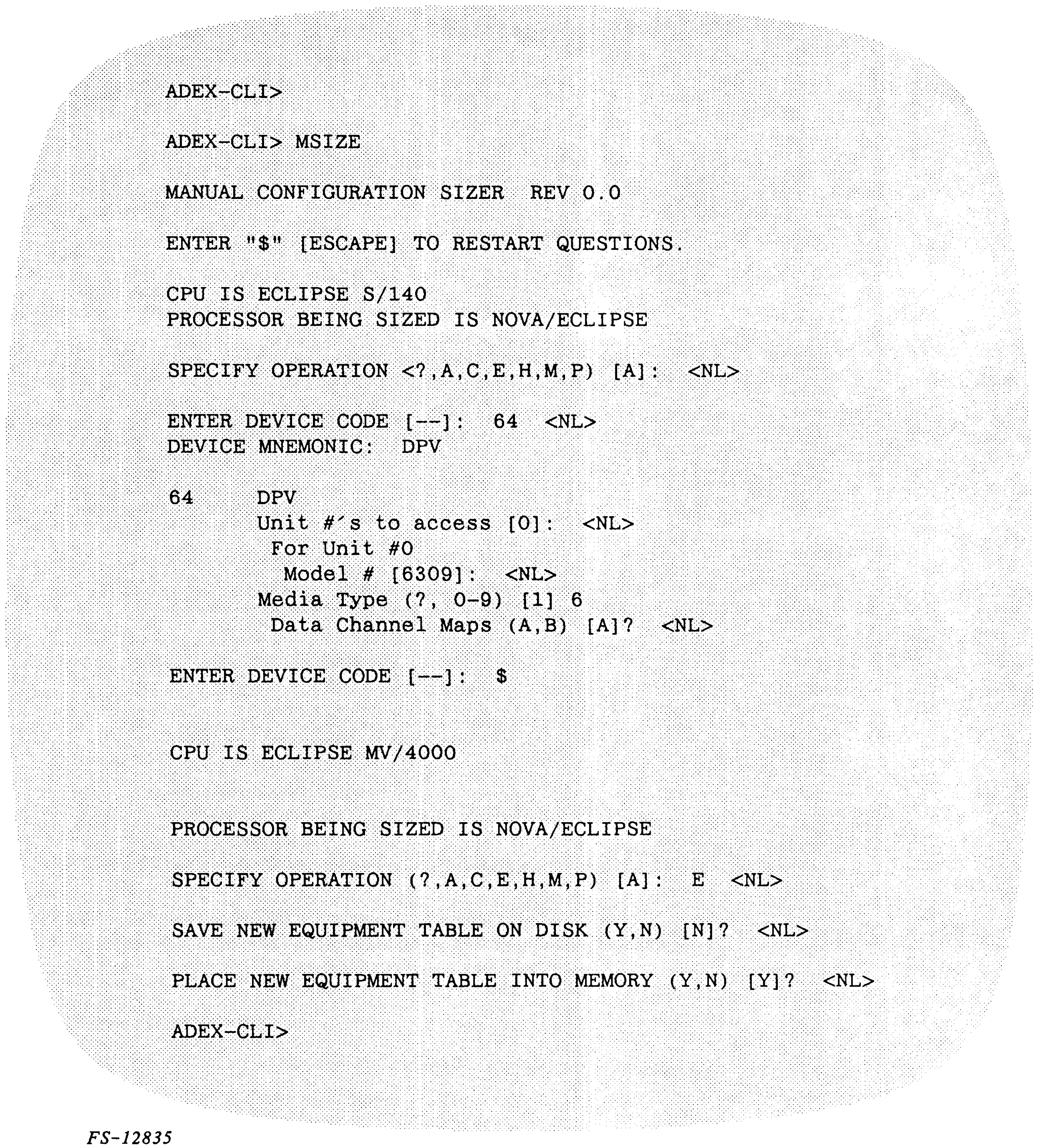
or

ADEX-CLI> **UPDATE.PRG**

The utility now asks several questions. All these questions have default answers.

CAUTION

Be sure to use only NEW LINE or Carriage Return to terminate your answers. An ESC will abort the current Update utility prompt and return you to CLI.



```
ADEX-CLI>

ADEX-CLI> MSIZE

MANUAL CONFIGURATION SIZER  REV 0.0

ENTER "$" [ESCAPE] TO RESTART QUESTIONS.

CPU IS ECLIPSE S/140
PROCESSOR BEING SIZED IS NOVA/ECLIPSE

SPECIFY OPERATION <?,A,C,E,H,M,P) [A]:  <NL>

ENTER DEVICE CODE [--]:  64  <NL>
DEVICE MNEMONIC:  DPV

64      DPV
        Unit #'s to access [0]:  <NL>
         For Unit #0
          Model # [6309]:  <NL>
        Media Type (?, 0-9) [1] 6
         Data Channel Maps (A,B) [A]?  <NL>

ENTER DEVICE CODE [--]:  $


CPU IS ECLIPSE MV/4000


PROCESSOR BEING SIZED IS NOVA/ECLIPSE

SPECIFY OPERATION (?,A,C,E,H,M,P) [A]:  E  <NL>

SAVE NEW EQUIPMENT TABLE ON DISK (Y,N) [N]?  <NL>

PLACE NEW EQUIPMENT TABLE INTO MEMORY (Y,N) [Y]?  <NL>

ADEX-CLI>
```

FS-12835

Figure 3-4. Listing an Update Tape in the System Equipment Table

The questions are as follows:

1. `DEVICE CODE [22]?`

   Enter the device code of the source media. The default is 22. The System Equipment Table must define at least one of the units associated with the device code as an ADEX update media (tape or disk based) or an ADEX system media (disk based). Otherwise, you will get the error message: EQUIPMENT NOT FOUND or NOT AN UPDATE MEDIA.

2. `UNIT NUMBER [x]`

   Enter the unit number of the source media. The System Equipment Table must define the unit specified as ADEX update media (tape or disk based) or ADEX system media (disk based). Otherwise, you the an error message: EQUIPMENT NOT FOUND or NOT AN UPDATE MEDIA> If the source media is a tape, you are asked question 3. Otherwise, you are asked question 4.

3. `FORMAT (0 = AOS  1 = ADEX) [0]?`

   Enter the format used for the source type.

4. `CONFIRM [N]?`

   If you take the default of N, Update just copies the update files to the runtime media. Otherwise, you must confirm the transfer of each file. Update will list the name of each file:

   filename.filetype [N]?

   and you must answer Y, or press NEW LINE to answer the default of N, to indicate whether or not to copy the file over.

5. `VERIFY [Y]?`

   If you answer Y, Update prints the name of each file it transfers to the runtime media. It includes the file type and the major/minor revision numbers. If you answer N, Update does not verify its actions.

6. `APPEND [N]?`

   If you answer N, Update transfers a file to the runtime media only if it already contains a file with that name . Question 7 is not asked.

   If you answer Y, Update transfers a file to the runtime media even if no copy of it exists on the runtime media.

7. `REPLACE [Y]?`

   If you decide to append files, Update now asks whether you want to replace them. When you answer Y, the Update utility replaces the files on the runtime media with the files of the same name on the source media. Question 8 will be asked.

   If you answer N, Update transfers files to the runtime media only if they do not already exist on it. Question 8 is not asked.

8. IGNORE REVS [N]?

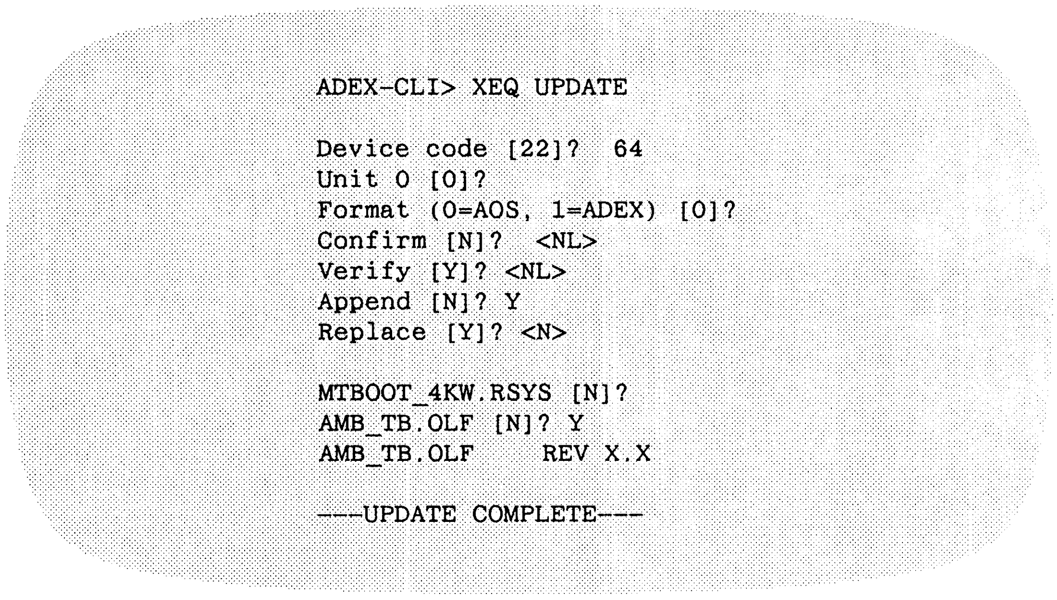
If you are replacing files, you have the option of ignoring major revision levels. When you answer N, Update only transfers files if the major revision level of the source file is higher than that of the file on the runtime media. When you answer Y, Update transfers files to the runtime media regardless of revision level.

Update now transfers files to the runtime media according to your instructions. When it finishes, you get the message:

```
--- UPDATE COMPLETE ---
```

and you return to the CLI prompt.

Figure 3-5 illustrates the Update interaction.



```
ADEX-CLI> XEQ UPDATE

Device code [22]?  64
Unit 0 [0]?
Format (0=AOS, 1=ADEX) [0]?
Confirm [N]?  <NL>
Verify [Y]? <NL>
Append [N]? Y
Replace [Y]? <N>

MTBOOT_4KW.RSYS [N]?
AMB_TB.OLF [N]? Y
AMB_TB.OLF     REV X.X

---UPDATE COMPLETE---
```

FS-12836

Figure 3-5. Update Interaction

3.5.3 Errors

If you get an EQUIPMENT NOT FOUND error message, you will return to ADEX-CLI. If the OPERATOR flag is off when the Update Utility is executed, an OPERATOR NOT PRESENT error is issued, and again, you will return to the CLI.

3.6 ADEX FILE EDITOR

The ADEX File Editor (FEDIT.PRG) lets you display and patch any file on the runtime disk (media), and lets you generate toggle files. FEDIT is used primarily to patch files which exceed 24 KW in length, making them unaccessible to the Edit Utility. FEDIT also allows you to enter and display data as numbers, instruction, or ASCII characters. When FEDIT is executed, you must first tell the utility which file to edit:

```
Pathname?
```

Both the file's name and type must be entered (e.g., MYFILE.UDF). If there is any problem opening the file, the system issues an error message and repeats the question. To return to ADEX-CLI at this point, press ESC.

If the file you requested exists, it is opened and can be edited. If the file does not exist and the file type is not TOGL?, the system issues an error message and repeats the question.

If you requested a TOGL? file (e.g., MYFILE.TOGL) that doesn't exist, FEDIT lets you create one. The following question is asked:

```
CREATE A NEW FILE (Y,N) [N]?
```

If you do not want to create a new file, answer N and the pathname question is repeated. If you do wish to create a new file, answer Y. The following question is asked:

```
FILE SIZE (# OF PAGES) [1]?
```

Enter the number of pages you want (one page equals 400 octal words). A toggle file of the specified size is created and then opened. Once the file is opened, you enter the main loop:

```
Mode (?,N,I,A) [m]? *

Address (0-xxx) [addr}? *

aaaaa xxxxxx yyyyyy<cr>
aaaaa xxxxxx yyyyyy<cr>

:  :

aaaaa xxxxxx yyyyyy<nl>
```

m is the current input/output mode (IOM). The three modes are:

- Numeric (N)
- Instruction (I)
- ASCII (A)

dadr is the default address, *aaaa* is the current address you are examining, and *xxxxxx* is the value at the current address, displayed according to the current IOM. To change the value, type in the new value (shown as *yyyyyy*), following the rules pertaining to the current IOM (see Table 3-8). Otherwise, just type a delimiter. The delimiter you type controls what the FEDIT does next. Table 3-8 summarizes the action each delimiter has.

Table 3-8. FEDIT Delimiter Actions

PROMPT	DELIMITER	ACTION
old value	<cr>	Close current location, open next location.
	<nl>	Close current location, return to the Address question.
	<esc>	Do not modify current location, return to Address question.
Address Question	<cr>	Open the address specified.
	<nl>	Open the address specified.
	<esc>	Return to the Mode question.
Mode Question	<cr>	Define the input/output mode and go to the Address question.
	<nl>	Define the input/output mode and go to the Address question.
	<esc>	Close the file and return to ADEX-CLI.

NOTE

The CTRL-C CTRL-A (^C^A) and CTRL-C CTRL-B (^C^B) abort sequences are disabled while FEDIT is running. This prevents the open file from being left in an unreadable state.

The current address is displayed in the current radix.

The input/output mode defines how the data at the current address is displayed and entered. The three modes and their syntax rules are:

1. *Numeric format* displays and inputs data as 16-bit numbers in the current radix.

2. *Instruction format* displays and inputs data as either NOVA® or ECLIPSE® instructions. The CPU type on which you are currently running determines how the data is interpreted. If you are running on a NOVA processor, all data is interpreted as strictly NOVA instructions. If you are running on an ECLIPSE processor, data is first checked to see if it is an ECLIPSE instruction.

   The following rules apply when in Instruction input/output mode:

   a. The last word in the file is always interpreted as a NOVA instruction. For example, in a four page file, if location 1777 contains 102070, it is interpreted as an ADCC# 0 0 instruction instead of an EJMP instruction.

   b. Only a NOVA instruction may be placed in the last word of the file, because FEDIT may not increase the size of a file.

c. The Eclipse MV 32-bit Instruction Set, the ECLIPSE EDIT instruction operations, the ECLIPSE Vector (VCT) and Load Effective Address (LEF) , and the NOVA Floating Point Instruction Set are not supported. The ECLIPSE Floating Point Instruction Set, ECLIPSE Standard Instruction Set, and NOVA Standard Instruction Set are supported.

d. ALC instructions have the following syntax:

mnemonic<carry><shift><#>acs acd

e. Memory reference instructions have the following format:

mnemonic<@>displacement<index>

If index is omitted, then displacement is treated as an address.

f. Memory accumulator instructions have the following format:

mnemonic ac<@>displacement<index>

If index is omitted, then displacement is treated as an address.

3. *ASCII format* displays and inputs data as one or two 8-bit pieces of data (bytes). Each byte can be entered as either an ASCII character or as a numeric value nit exceeding 377 (octal). To enter a numeric value, enclose the number in angle brackets (e.g., <32>). Valid examples of ASCII format are:

AB	Left byte = "A"	–	Right byte = "B"
A<16>	Left byte = "A"	–	Right byte = "16"
<74>x	Left byte = "74"	–	Right byte = "x"
<16><33>	Left byte = "16"	–	Right byte = "33"
v	Left byte = "0"	–	Right byte = "v"
<14>	Left byte = "0"	–	Right byte = "14"
A<0>	Left byte = "A"	–	Right byte = "0"
61	Left byte = "6"	–	Right byte = "1"

When the file is closed, a new checksum is calculated for the file so that subsequent reads of the file will not return file checksum errors. The message *filename.filetype* UPDATED verifies that the file has been closed. The program then returns to ADEX-CLI.

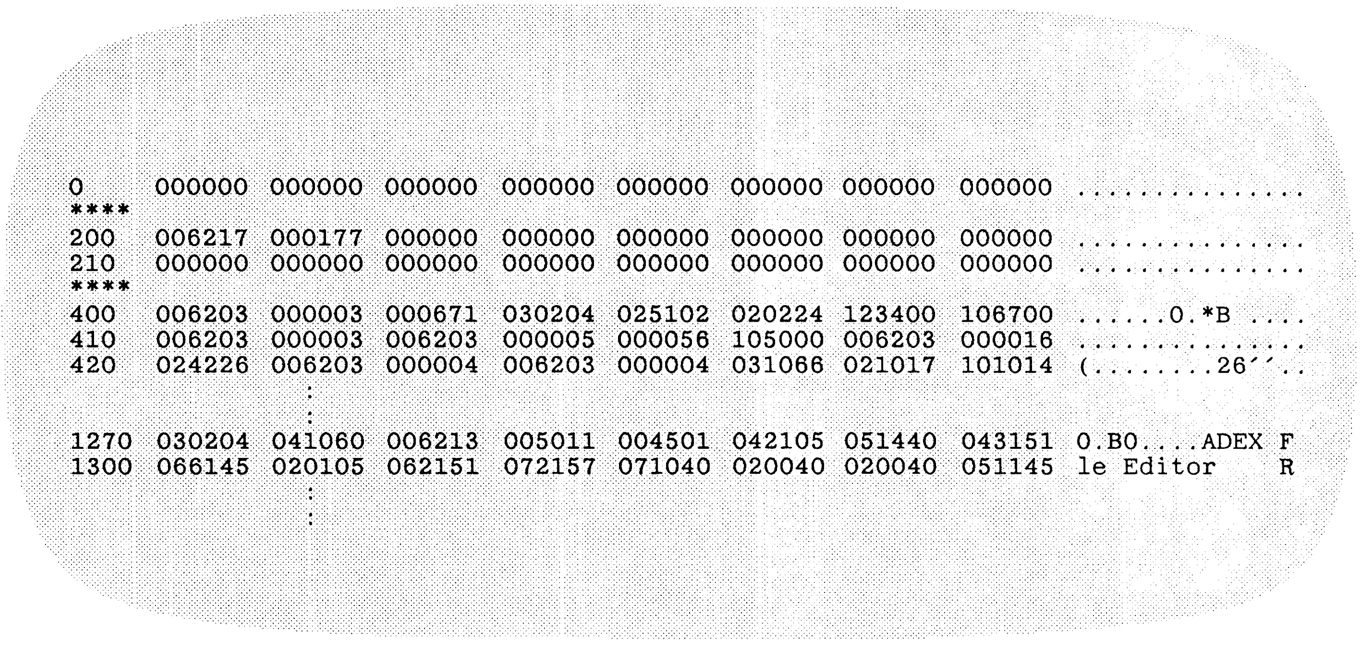
3.7 THE FILE DISPLAY UTILITY

The ADEX File Display Utility (DISPLAY.PRG) lets you display the contents of a file in both numeric and ASCII format. The numeric form is printed in the current radix. When DISPLAY is executed, you must tell the utility which file to display:

```
        Pathname?
```

Both the file's name and type (e.g., MYFILE.UDF)must be entered. If there is any problem opening the file, the system issues an error and repeats the question. To return to ADEX-CLI at this point, press ESC.

DISPLAY then reads in and displays the contents of the file in 4 KW segments. Each line of the display represents 8 words of the file, listed in order from left to right. Column 1 is the address of the first word listed on the line. Columns 2–9 contain the numeric contents of the 8 words, and column 10 (the rightmost column) is the 16 ASCII character equivalent of the 8 words listed in columns 2–9 Figure 3–6 shows an example of the contents of a file.



```
0    000000 000000 000000 000000 000000 000000 000000 000000 ................
****
200  006217 000177 000000 000000 000000 000000 000000 000000 ................
210  000000 000000 000000 000000 000000 000000 000000 000000 ................
****
400  006203 000003 000671 030204 025102 020224 123400 106700 ......0.*B ....
410  006203 000003 006203 000005 000056 105000 006203 000016 ................
420  024226 006203 000004 006203 000004 031066 021017 101014 (.........26´´..
                :
                :
1270 030204 041060 006213 005011 004501 042105 051440 043151 0.BO....ADEX F
1300 066145 020105 062151 072157 071040 020040 020040 051145 le Editor     R
                :
                :
```

FS-12837

Figure 3–6. Contents of a File

"8888" in the left margin means that the locations not displayed at that point all contain zeros. Bytes containing values less than 40 octal, or RUBOUT characters (177), are displayed as "." in the ASCII format column.

3.8 DTR FORM PRINT UTILITY

The DTR Form Print Utility (DTRFORMS.PRG) lets you print one or more DTR forms at one time on the system console device. If a secondary output device is defined prior to calling the utility, the DTR forms will be printed on the secondary output device. To execute the utility, type:

ADEX–CLI> **DTRFORMS** *x*

where *x* is the number of forms you want to print.

When the forms are finished printing, the message, `DTR FORMS PRINTED` is displayed.

3.9 CONTIGUOUS MEMOTY TEST UTILITY (CMEMT)

The Contiguous Test program (CMEMT.PRG) verifies that each 1 KW in memory exists. The tested memory range is from location 0 to the top of the physical memory. Top of physical memory is derived from the contents of system variable PMSZH?, located in Pseudo Page 0. The value of PMSZH? corresponds to the value of Word 2 in the System Equipment Table. Both values are defined at System Initialization and are modified when the Manual Sizer's "M" operation is executed or when a new System Equipment Table is read in via the **EQUIPMENT READ** ADEX CLI command.

If the Operator flag is ON, a banner message is printed. If no non-existent memory is detected (i.e., if memory is contiguous), a verification message is printed. If the Operator flag is OFF, the only messages printed are those describing any detected non-existent memory.

The KW numbers printed refer to the address of that KW and are printed in decimal. For example, in a 128 KW system, the first KW is 0, and the last KW is 127.

No SWREG bits are used by CMEMT.

3.10 THE SYMBOLIC DEBUGGER (ADEB)

The Symbolic Debugger (ADEB) lets you debug programs by accessing and changing the contents of any internal register or memory location. You can do this by controlling a program's execution with breakpoints (which stop the program at specific points). By placing breakpoints at appropriate places in a program, you can monitor a test program's progress and examine the operation of subtests. The Debugger lets you insert up to eight different breakpoints in a program. After a test program halts at a breakpoint, you can examine or change the values in memory or in certain registers. Then, you can restart the test program. Breakpoints defined with ADEB are maintained until you instruct the ADEB to remove them.

When you run the Debugger, the CLI loads it first, followed by the ADEX test program. If the program is too long to fit, you will not be able to run it with the Debugger. You will get the message MEMORY RESOURCES NOT AVAILABLE. ADEB requires 2.5 KW. On a 16 KW machine, you have 5.5 KW available to use. With a 32KW machine, you have 21.5 KW available.

The Debugger uses locations 50–57 in memory. Make sure your test programs do not use these locations.

To execute the Debugger, type:

ADEX-CLI> **DEbug** *filename[.PRG] [start_address]*

where *filename* is the name of the program you are loading. If you provide an extension other than .PRG, the CLI issues an error message. *start_address* is the optional starting address for the Debugger. If you do not provide a starting address, the Debugger is loaded at the top of user space in memory.

For example:

ADEX-CLI> **DEBUG** *PROG1.PRG*

loads the Debugger at the top of user space in memory and loads *PROG1* at location 0.

The CLI then prints the starting address of the Debugger and starts it. Commands must be in uppercase. ADEB does not recognize lowercase letters.

Table 3-9 summarizes the basic Debugger commands. For a more complete discussion, refer to the Symbolic Debugger User's Manual (093-000044) or the ECLIPSE Symbolic Debugger (093-000140).

NOTE

The Debugger displays a ? or U when you use an invalid command.

Table 3-9. Debugger Commands

OPERATION	COMMAND	FUNCTION
Display/change memory cells	adr! adr/	Open location adr and print contents.
	CR	Close open location and open next one.
	LF	Close open location.
	^	Close open location and open preceding one.
Display/change registers	$A	Display contents of all accumulators.
	n$A	Open register n (n = 0-3 for the NOVA computer; 0-7 for the ECLIPSE computer).
Display/set breakpoints	$B	Display locations of all set program breakpoints.
	adr$B	Insert breakpoint at location adr.
Delete breakpoints	$D n$D	Delete all breakpoints n (n = 0-7).
Restart program execution	$P	Restart execution from a breakpoint with break proceed counter set to +1 (executes breakpoint once before entering debugger program).
	n$P	Restart execution from a breakpoint with break proceed counter set to counter n (n = 0-7).
	n$Q	Open break proceed counter n (n = 0-7).
	addr$R	Restart execution at adr with I/O reset.
Select output mode	=	Display output in numeric format.
(use as terminator for input or after display of cell contents)	:	Display output in symbolic format.
	;	Display output in instruction format.
	<-	Display output in half-word format.
	'	Display output in ASCII format.
	&	Display output in byte pointer format.
Set output mode	$=	Display future output in numeric format.
	$:	Display future output in symbolic format.

Table 3-9. Debugger Commands (continued)

OPERATION	COMMAND	FUNCTION
Set output mode (continued)	$;	Display future output in instruction format.
	$<-	Display future output in half-word format.
	$'	Display future output in ASCII format.
	$&	Display future output in byte pointer format.
Search memory (press any key to stop search)	$S	Search all memory.
	adr$S	Search memory from location 0 to adr.
	adr<$S	Search memory from adr to memory limit.
	adr1<adr2$S	Search memory from adr1 to adr2.
Display/change	$C	Open carry/teletype output register.
Special registers	$F $L $M $N $W	Open Floating Point registers*. Open Location registers. Open Mask register. Open Number register. Open Word register.
* Applies to systems with floating point option.		

APPENDIX A
GLOSSARY

APPENDIX A
GLOSSARY

ADEX Link Program (ADESL): Program supplied in separate file on ADEX tape. It is loaded into the root directory to provide a link to the diagnostic area of CORESIDENT disks.

Alignment Programs: Test programs that help align the heads of magnetic recording devices.

Auto Sizer (ASIZE): Program that determines the system configuration and places the information in the System Equipment Table. Requires no operator input.

Bootstrapping: The process of loading ADEX from tape or disk into memory.

Breakpoint: Octal debugger command designed with a "B" that terminates a program with a certain address that has a "B" designated. It returns to the debugger program.

Checksum Value: Verifies the integrity of the memory-resident operating system code by summing all the memory-resident code and comparing the total to a known value.

CONLOG RSYS: On disk media, the name of the disk file in which the console log is saved.

CORESIDENT: ADEX and your operating system existing on the same disk. However, only ADEX or your operating system may be operating at one time.

Diagnostic Programs: ADEX test programs that detect and isolate hard faults on subsystems and modules. They also help to isolate a problem to the failing circuit by providing functions such as looping on an error.

Double Keystrokes: A simple way to complement a flag in the Environment Control Word. When you have a prompt or when a program is awaiting input, you use CTRL-P plus the appropriate single keystroke command associated with the flag you wish to change.

DRVR File: Driver file that contains executable code that calculates and performs I/O operations to peripherals supported by ADEX.

DTOS: An earlier diagnostic operating system superceded by ADEX. It is still used with some products. DTOS commands are not compatible with the ADEX feature.

Edit Utility: Serves as a text handling and patch utility. Lets you edit any type of existing file and create source, text, or toggle files.

Environment Control Word: A word in memory that lets you control many aspects of the ADEX operating system.

EQUIPO.UDF: A default disk file to which you can save the information from the System Equipment Table.

Formatter Programs: ADEX test programs which initialize magnetic disk media.

Initialization Monitor: Program that takes over when you load ADEX to initialize some system variables.

Kernel: The portion of the memory-resident system code that enables ADEX to perform all of its system functions and utilities.

Logging: The process by which system console output is saved either in the disk file CONLOG.RSYS or on a log tape.

Logical Disk Unit (LDU): It may consist of a single physical disk or multiple disks.

Macro: A file that contains a sequence of ADEX-CLI commands that execute automatically when you type in the name of the file. Also referred to as a macro or SRCE (source) file.

Manual Sizer (MSIZE): A utility program that lets you manually add or modify the information in the System Equipment Table.

Octal Debugging Tool (ODT): Provides user with limited debugging facilities without using any of the ADEX system routines. Uses one self-deleting breakpoint.

OLF file: A section of a larger program. Large programs which will not fit into memory are written in sections. The root (.PRG) section executes in memory first and the other sections (.OLF) are called into memory when needed.

PRG File: A program that runs under ADEX. It is started at location 200 (8).

RSYS File: A system program that remains memory-resident once it is invoked.

Runtime Media: The tape or disk containing ADEX from which you are currently executing.

Screen Edit: A utility used when running on a video console that allows you to make changes to a command line before you execute it and/or to use the previous command line in forming the next one.

Script Builder: A program that lets you produce a script file to be saved on runtime media, or a one page script that is executed and then discarded.

Script: A file that contains a sequence of ADEX-CLI commands that execute automatically when you type in the name of the file. Also referred to as a macro or SRCE (source) file.

Single Keystrokes: Single digits or alpha characters used while a program is running to complement the meaning of any flag in the Environment Control Word.

Sizing: A process whereby ADEX identifies all the hardware in the system to produce the System Equipment Table (SET).

SNRU File: A System Nonresident Utility file. It contains executable code (the code that is executed when an ADEX system call is made).

SOP File: Standalone program that is executable by typing in the program name in response to the FILENAME [ADEX]? bootstrapping question. Standalone programs do not use any of the ADEX system utilities. SOP files are started indirectly through location 2. SOP files do not pertain to MV ADEX.

SRCE File: A file that contains a sequence of ADEX-CLI commands that execute automatically when you type in the name of the file. Also referred to as a macro or script.

SRCH File: A SEARCH test file (System Exerciser and Reliability CHeck) that is 2 KW in length and is written contiguously on disk media. (NOVA/ECLIPSE and microECLIPSE ADEX only.)

String: A temporary storage place that holds up to 80 characters. It is used to pass textual information to a program.

Switch Register: Two 16-bit registers used for passing numerical information to ADEX programs.

Symbolic Debugger (ADEB): Allows you to debug programs by accessing and changing the contents of any internal register or memory location. Uses 8 non-self-deleting breakpoints.

System Equipment Table (SET): A table in memory which lists all the peripherals and options in the system.

System Exerciser Programs: ADEX test programs which detect and isolate hard, intermittent and interactive faults in systems and distributed systems. Their error reports indicate either which subsystem or FRU may be failing.

System Identification: An 80-character string that identifies the ADEX software on the system media. The SYSID command displays the identification message. If the runtime media is disk, you may use the command to change the message.

System Initialization Program (SINIT): Takes over (if the Initialization Monitor encounters no problems) and concludes the power-up sequence,

System Media: A tape or disk containing ADEX.

System Panic: A fatal ADEX error. Refer to Appendix C for more information.

Templates: Characters (* and +) that serve as a shortcut in referring to filenames.

Timing Programs: ADEX test programs which provide help in calibrating equipment to meet timing specifications.

TOGL File: An operator-generated file, created through the Edit utility that performs a specific function.

TXT File: An ASCII text file used for any purpose.

Update Media: Diskette media used to update runtime media, which must be disk. This media can be built using the AOS **COPY** command.

Update Utility: Lets you update an ADEX system disk without rebuilding it.

Utility Program: A program which allows you to perform a standard task. Some examples are building more ADEX media, sizing the system, editing a file, etc.

Verification Programs: ADEX test programs which prove that a product performs according to its functional specification.

Write-protected Media: A disk or tape that cannot be written to.

APPENDIX B
BOOTSTRAP TROUBLESHOOTING

APPENDIX B
BOOTSTRAP TROUBLESHOOTING

If the bootstrapping process is successful, and the system hardware passes the test given by the ADEX bootstrap program, the following message appears:

```
NOVA INSTR OK
0-377 MEMORY OK
4000-TOP MEMORY OK
```

followed with, "FILENAME [ADEX]", appears on NOVA/ECLIPSE (NE), microNOVA (MN), and microECLIPSE (ME) ADEX. The ADEX Menu appears on MV ADEX (MV). If the test fails, the CPU is halted before the words are completely displayed. This appendix is a guide to help you find the problem. Table B-1 lists some probable causes depending on the output of the bootstrap message.

Table B-1. Bootstrap Message Summary

BOOTSTRAP MESSAGE	CAUSES (in priority)
Nothing	Wrong media/density Boot device failure Faulty console device Memory failure CPU failure
NOVA INSTR OK	Lower memory failure
0-377 MEMORY OK	Upper memory failure
4000-TOP MEMORY OK	Passed bootstrap test

Refer to Table B-2 for a listing of bootstrap tasks and their associated sequence numbers.

Table B-2. Bootstrap Task Summary for Tape and Disk Systems

SEQUENCE	TASK INITIATED	
1	NOVA Instruction Test	
2	Print "NOVA INSTR OK"	
3	Test memory locations 400-3777	(disk only)*
4	Load Multisector Loader (MSL)	(disk only)
5	Checksum MSL	(disk only)
6	Size the system console	
7	Test memory locations 0-377	
8	Print "0-377 MEMORY OK"	
9	Test memory locations 4000-top of memory	
10	Print "4000-TOP MEMORY OK"	

* Disk only, with the following exceptions: all current and future UNICORN disks, e.g.: Models 6236, 6239, 6310, 6328, 6329, 6363, etc.

Errors that occur after you select one of the menu options in MV ADEX or answer the FILENAME question in NE/MN/ME versions of ADEX, will result in one of three error messages. The comments that follow the error message explain what could have caused it and possible solutions.

1. `FILE DOES NOT EXIST`

   If you selected ADEX by either selecting Option 5 of MV ADEX or typing NEW LINE (NE, MN, ME ADEX), then the boot program could not find either the KERNEL program or the runtime media driver. If you selected a stand-alone program, the file you specified does not exist on the media.

2. `FILE CHECKSUM ERROR`

   The file just loaded did not checksum correctly. This could be caused by bad media or memory problems. Try again. If the problem still exists, then rebuild the media. If the problem still persists, try another media. If that does not solve the problem, then the cause is probably system-related (such as memory, drive, or controller).

3. `DEVICE STATUS ERROR, DIA = xxxxxx`

   The boot program driver received a DIA status error. *xxxxxx* is the octal value of the DIA register received.

APPENDIX C
PANIC CODES

APPENDIX C
PANIC CODES

AN ADEX panic occurs when an unrecoverable error is detected. The following message appears:

```
        FATAL ADEX ERROR # xxxxxxx
```

xxxxxxx is the octal panic code. After the panic message, control is transferred to the ODT. The contents of the accumulators and carry are printed, and an ODT prompt is issued. Refer to Table C–1 for a listing of panic codes and their meanings.

To continue after a panic has occurred, use the ODT's P command. Depending on the severity of the error, the panic recovery routine may or may not work. If it does, you will get an **ABORT** message, followed by the ADEX–CLI prompt. If the system doesn't recover, then manually reboot the runtime media.

On panics where a file was unable to be read in, AC0 contains the error code that describes the reason for read failure. Use the **ERMES** command for error code explanations.

Table C–1. Software Panic Codes

PANIC CODE	DESCRIPTION
0	Jump to 0.
1	Stack underflow.
2	Stack underflow.
3	Unable to access the teletype input package.
4	Fatal error return from Console Log Driver (CONLOG?). ACO = error code.
5	Unable to access the CLI utility (file CLI.SNRU).
6	Unable to access the filename of standard driver for system console.
7	Unable to read in system console driver file.
10	Unable to access CLIEER.AC0 = error code.
11	Attempt to output a character with system console driver undefined.
12	Unable to access abort module during an abort.
13	Kernel checksum error during 4 KW to 8 KW Kernel restoration.
14	System console or secondary output device timeout.
15	Unable to read in System Initialization program (SINIT).

Table C-1. Software Panic Codes (continued)

PANIC CODE	DESCRIPTION
16	Unable to read in PBL while trying to abort or terminate a script.
17	Unable to write out PBL while trying to abort or terminate a script.
20	SCALL? 0 attempted.
21	I/O error occurred before the system console device driver could be put into place. AC0 = device DIA status. AC1 + IOCB status (described in Table C-2).
22	Unknown model number received from ECLID instruction during CPU sizing.
23	Unable to load the CPU/Console/Memory sizer program CCMSZ.
24	Fatal error encountered within ERRMSG utility. AC0 = error code.
25	Unable to access EXCCHR within system console driver. AC0 = error code.
26	Duplicate filename found in directory structure.
27	Unable to access CLI support routine (file CLISR).
30	Kernel checksum failure during an abort.
31	Unable to read in the runtime media's Master Disk Directory.
32	Unable to access control characters routine (file EXCCHR).
33	Unable to relocate RSYS back to the top of logical memory.
34	DIA Status error received during a memory dump.
35	Directory segment integrity lost.
36	Device error from system console, or its slave processor or controller.
37	Unable to release file's memory resources after a file read error.
40	Unable to access terminate routines (file TERMINATE).
41	Unable to access Screen Edit utility (file SEDIT).
42	Jump indirect through location 1. (Interrupts illegally enabled).
43	Unable to resolve Kernel.RSYS during 4 KW to 8 KW Kernel restoration.
44	Nonresident system call made when caller was not at stack base level during 4 KW mode.
45	Unable to find secondary output device driver filename during 4 KW to 8 KW Kernel restoration.
46	Unable to read in secondary output device file during 4 KW to 8 KW Kernel restoration.
47	Unable to resolve runtime media driver during 4 KW to 8 KW Kernel restoration.
50	Runtime media driver checksum error during 4 KW to 8 KW Kernel restoration.
51	RSYS overwrite attempt while in 4 KW mode.

Table C-1. Software Panic Codes (continued)

PANIC CODE	DESCRIPTION
52	MPT mini-diskette read or write operation attempted with MPT mini-diskette 512-word buffer undefined.
53	Unable to read in Kernel program during a media switch.
54	Unable to initialize runtime media after an IORST.
55	Attempt to execute a 32-bit system call, utility, or initialization routine on a 16-bit processor.
56	32-bit instruction or extended memory test failed.
57	DOIO called executed with AC3 as source or destination accumulator.
60	Unable to access "GTSF.SNRU". AC0 = error code.

The IOCB (I/O Command Block) status word is formatted, as shown in Table C-2.

Table C-2. IOCB Status Word

BIT	DESCRIPTION
Bit 0	1 = IOCB active
Bits 1-4	Not used.
Bits 5-7	Error Number 0 = DIA status error (DIC for fixed head disks) 1 = DIB status error 2 = BMC error (BMC status cleared) 3 = Controller full timeout 4 = Operation timeout 5 = Invalid model number 6 = Invalid command
Bits 8-13	Not used.
Bits 14-15	1 = done 3 = done + error

APPENDIX D
MNEMONICS DESCRIPTIONS

APPENDIX D
MNEMONICS DESCRIPTIONS

Refer to the mnemonics HELP files for the latest information. To do so, type HELP MNEMONICS. Table D-1 lists the Mnemonic Codes and their descriptions.

Table D-1. Mnemonic Codes and Descriptions

MNEMONIC	NOTE	MODEL #	DESCRIPTION
ADCV	G,1	4120	A/D Converter
		4130	A/D Converter
		4140	A/D Converter
		4150	A/D Converter
		4223	A/D Converter (Micro Products)
ALM	C	4255	ALM/4,8,16
		4227	Async Interface (Micro Products)
AP	G,1	8620	Array Processor (AP130)
		8644	Array Processor (S/250)
		8661	Array Processor (IOP)
AP2000	G	5520	Array Processor 2000.
ATP	G,3	0	Attached Processor (DG10)
		8902	Attached Processor (DG45)
IOC	G	9999	INTEGRATED I/O CONTROLLER
BMC	G	8642	High Speed (Burst Multiplexor) Channel (S/250, C/350, M600)
		8699	High Speed (Burst Multiplexor) Channel (S/140)
		8734	High Speed (Burst Multiplexor) Channel (S/20)
		8772	High Speed (Burst Multiplexor) Channel (S/280)
BMCT	G	9642	BMC Tester
BPC	G,1	4348	Bit Synchronous Controller, 1 line
		4349	Bit Synchronous Controller, 4 lines
CDR	G	4016	Card Reader

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
CHAR	G	8614	Character Instruction Set (S/130)
		8639	Character Instruction Set (S/250)
		8664	Character Instruction Set (S/140)
		8731	Character Instruction Set (S/120)
CIAC	C	4543	Combat Intelligent Async. Controller
CISC	C	4543	Combat Intelligent Sync. Controller
COMM	G	8633	Commercial Instruction Set
CPI	G	4398	Diamond PBX Communications Controller
CRC	G,1	4228	CRC Gen/Chk (Micro Products)
		4266	CRC Gen/Chk
DACV	G,1	4180	D/A Converter
		4224	D/A Converter (Micro Products)
		4300	DG/DAC
DCU	G	4250	Data Control Unit DCU/50
		4254	Data Control Unit DCU/200
DIF	G,1	4335	Digital Interface Card
DIO	G,1	4065	Digital I/O
		4222	Digital I/O (Micro Products)
DKB	D	6063	1 MB Fixed Head Disk
		6064	2 MB Fixed Head Disk
DKM	D,4,5	8323	358 KB Mini-diskette (MPT series)
DKP	D	4047	2.5 MB Diablo Disk
		4048	6 MB Century Disk
		4057	25 MB Century Disk
		4231	92 MB CDC Disk
		6030	315 KB Floppy
	2	6045	10 MB Phoenix Disk
	2	6070	20 MB Gemini Disk
		6097	1.2 MB Quad Density Floppy
		6099	12.5 MB Echo Disk
		6103	25 MB Echo Disk
		6225	5 MB Cactus Disk
		6227	15 MB Cactus Disk
		6234	50 MB Daisy Disk
	2	6095	10 MB Phoenix Disk (Micro Products)
DKT	D,4	6038	315 MB Floppy (Micro Products)
DPB	D	6224	15 MB BMC Cactus Disk (Micro Products)
		6280	50 MB BMC Daisy Disk (Micro Products)

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
DPF	D	6060	96 MB Zebra Disk
		6061	190 MB Zebra Disk
		6067	50 MB Zebra Disk
		6122	277 MB Zebra Disk
		6160	73 MB Kismet Disk
		6161	147 MB Kismet Disk
		6214	602 MB Kismet Disk
DPM	G	8632	Demand Paging Map
DPV	D,4	6309	368,737 KB Gnome I, II Mini-diskette
		6310	38 MB Harem II Winchester Disk
		6236	354 MB Argus Disk
		6239	592 MB Argus II
		6328	71 MB Harem III Winchester Disk
		6329	120 MB Harem III Winchester Disk
		6357	900 MB Argus III
		6363	160 MB Harem IV Winchester Disk
DRTC	G,3	0	DCU Real Time Clock
DSK	D	4514	368 KB Mini-diskette
		6096	1.2 MB Quad Density Floppy (Micro Products)
		6102	12.5 Echo Disk (Micro Products)
		6105	25 MB Echo Disk (Micro Products)
		6220	5 MB Cactus Disk (Micro Products)
		6222	15 MB Cactus Disk (Micro Products)
		6271	15 MB Harem I Winchester Disk (DG)
		6267	368 KB Mini-diskette (DG10)
		6268	368 KB Mini-diskette (DG20, DG30)
		6301	38 MB Harem II Winchester Disk (DG,Obsolete)
		6302	737 KB Gnome II Mini-diskette (DG10)
		6303	737 KB Gnome II Mini-diskette (DG20, DG30)
DSK	D	9998	120 MB Harem III Winchester Disk (Micro Products)
		6336	71 MB Harem III Winchester Disk (Micro Products)
ERCC	G	8400	Error Correction (S/230, C/330, S/200, C/300)
		8612	Error Correction (S/130, AP130, C/150, S/250, C/350, M600)
		8678	Error Correction (S/140)
FPU	G	8413	Floating Point (EAU)
		8539	Floating Point (NOVA 3)

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
		8613	Floating Point (S/130)
		8622	Floating Point (C/150)
		8641	Floating Point (CPU 3,4)
		8662	Floating Point (S/140 Firmware)
		8663	Floating Point (S/140 Hardware)
		8731	Floating Point (S/120, S/20, DG)
GDE	G,1	4436	Graphics Data Entry Mouse (DG)
		4437	Graphics Data Entry Tablet (DG)
GRA	G	2718-B	Graphics Controller DS/7500 – 2 or 8 bits/pixel
		2719	Graphics Controller DS/7500 – 8 bits/pixel
		2717-A	Graphics Controller DS/7500 – 2 bits/pixel
		2717-B	Graphics Controller DS/7500 – 8 bits/pixel
		2724	Graphics Controller DS/7500 – 8 bits/pixel
GRX	G	8802	Graphics Controller (Blitz) low res 1 mem card
		8804	Graphics Controller (Blitz) low res 3 mem cards
		8807	Graphics Controller (Blitz) high res 1 mem card
		8806	Graphics Controller (Blitz) high res 3 mem cards
HHC	G,1	8564	microNOVA Hand Held Console
HOST	G,3	0	DCU Host Interface
HSC	G,1	4229	microNOVA High Speed Channel
IAC	C	4357	Intelligent Asynchronous Controller, 8 lines
		4358	Intelligent Asynchronous Controller, 16 lines
		4367	Octbat Intelligent Async. Controller, 8 lines
IAC	C	4368	Hexbat Intelligent Async. Controller, 16 lines
IBC	G	4555	Intelligent Broad Ban Lan Controller for MV Systems
IDC	G	4581	LAN Controller
IVC	G	4550	Parrot Intelligent Voice Controller

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
ILC	G	4532	Intelligent LAN Controller (Ethernet IEEE 802)
IOP	G	8660	I/O Processor
IOPT	G,3	0	I/O Processor Timer
IOT	G	8000	I/O Tester
		8001	I/O Tester (Micro Products)
IPB	G,1	4240	Inter Processor Buss
ISC	C	4380	Intelligent Synchronous Controller, 2 lines
		4530	Programmable Synchronous Controller, (M.P.), 2 lines
LAN_RX	G	----	Interlan and Aardvark LAN Receiver
LAN_TX	G	----	Interlan and Aardvark LAN Transmitter.
LPT	G	4034	240-300LPM
		6043	TP1 Printer
		6086	LP2
		6088	LP2 DCH Printer
		4216	240-900LPM DCH
		4320	Letter Quality
		4323	Band Printer
		6216	Jalapeno 180 CPS Printer
LAC		4560	MV/2000DC DS/7500 Asynch. Controller
		5560	
LDC		4580	MV/2000DC DS/7500 LAN
LLC		4562	MV/2000DC DS/7500 LAN
LSC		4561	MV/2000DC DS/7500 Synch. Controller
MBC	G	5220	D280C Color Display
		6012	DGC Display
		6040	Hardcopy DASHER, Infoton Display
		6052	LCD, Upper Case
		6053	LCD, Upper/Lower Case
		6106	D100/D200 Display
		6130	D400/D450 Display
		6150	G300 Graphics Display Terminal
		6166	D410 Display Workstation
		6167	D460 Display Workstation
		6168	D210 Display Terminal
		6169	D211 DIsplay Terminal

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
MBC	G	6241	G500 Color Graphics Display Terminal
MCAT/R	G	4206	MCA Transmitter/Receiver
MDV	G	8534	Multiply Divide Option
MLNC	G,1	4529	IEEE-802 LAN Controller (Micro Products)
MLPT	G	8690	MPT Series PIO Lineprinter
MMPU	G	8394	NOVA 4 MAP
		8412	ECLIPSE (S/200, C/300)
MMPU1	G	8618	ECLIPSE MAP (Other than S/200, C/300, S/140)
		8678	ECLIPSE MAP (S/140)
MMU	G	8535	NOVA 3 MAP (no protection)
MODEM	G	----	Modem Interface on Aardvark
MOUSE	G	----	Mouse Interface on Aardvark
MPT	G	8321	MPT/80, MPT/83, MPT/87 Console Interface
		8322	MPT/100 Console Interface
MPU	G	8538	NOVA 3 MAP
MRTC	G,3	0	microNOVA RTC
MTA	T	4307	Dual Mode GCR
		6020	Standard Tape (*00 BPI)
		6026	Dual Mode Tape
		6123	Streamer Tape (Micro Products)
		6125	Streamer Tape (NOVA/ECLIPSE)
		6230	Oasis Cartridge Tape Subsystem (Micro Products)
		6231	Oasis Cartridge Tape Subsystem (NOVA/ECLIPSE)
		6270	Palmtree Cartridge Tape Subsystem (DG)
		6300	6250 BPI High Speed Tape Subsystem
		6311	Palmtree Cartridge Tape Subsystem (Aardvark)
MTJ	T	6341	Unicorn 25 MB Reel to Reel
		6351	Floppy Tape (MV/2000DC)
		6352	Unicorn 67 MB Cartridge Tape
NBA	G	4460	Network Bus Adapter
NVM	G,1	8316	Non-Volatile Memory (Micro Products)

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
PAR	G	8536	Parity Option
	3	0	Parity Unit (ECLIPSE S/20)
PIT	G	4217	Programmable Interval Timer
PLT	G,1	4017	Plotter
		4435	Graphics Plotter
PROMB	G,1	8574	PROM Burner (Micro Products)
PTP	G	4012	Paper Tape Punch
PTR	G,1	6013	Paper Tape Reader
QTY	C	4060	Quad MUX
RTC	G	4008	Real Time Clock
SDX	G,1	8320	MBC/SDX I/O Board
SLM	C	4263	SLM/2
		4226	Sync Interface (Micro Products)
TLC	G,1	4516	Talker/Listener Controller (Micro Products)
		4517	Talker/Listener Controller (NOVA/ECLIPSE)
TTI	G	5220	D280C Color Display
		6012	DGC Display
		6040	Hardcopy DASHER, Infoton Display
		6052	LCD, Upper Case
		6053	LCD, Upper/Lower Case
		6106	D100/D200 Display
		6130	D400/D450 Display
		6150	G300 Graphics Display Terminal
		6166	D410 Display Workstation
		6167	D460 Display Workstation
		6168	D210 Display Terminal
		6169	D211 Display Terminal
		6241	G500 Color Graphics Display Terminal
		6265	Color Graphics Display (DG)
TTO	G	4433	136 Column, 160 CPS Dot Matrix Printer (DG)
		4434	80 Column, 160 CPS Dot Matrix Printer (DG)
		4518	136 Column, 35 CPS Letter Quality Printer (DG)
		4531	Wide Wonder Printer
		5220	D280C Color Display
		6012	DGC Display

Table D-1. Mnemonic Codes and Descriptions (continued)

MNEMONIC	NOTE	MODEL #	DESCRIPTION
		6040	Hardcopy DASHER, Infoton Display
		6052	LCD, Upper Case
		6053	LCD, Upper/Lower Case
		6106	D100/D200 Display
		6130	D400/D450 Display
		6150	G300 Graphics Display Terminal
		6166	D410 Display Workstation
		6167	D460 Display Workstation
		6168	D210 Display Terminal
		6169	D211 Display Terminal
		6215	D240 (Jalapeno) Display Terminal
		6241	G500 Color Graphics Display Terminal
		6242	D210 Display Terminal
		6243	D240 Display Terminal
		6255	D460 Display Terminal
		6256	D460 Display Terminal
		6265	Color Graphics Display (DG)
		6284	D410 Display Terminal
		6308	D470C Display Terminal
UASYNC	G	9999	Unicorn Sync/Async Interface
UCONV	G	9999	Unicorn Converter
UGRAPH	G	9999	Unicorn Graphics
ULAN	G	9999	Unicorn Local Area Network
ULM	C	4243	ULM5 Sync/Async Line Multiplexor
		4342	ATI-16 Asynchronous Terminal Interface
	1	4336	Async/Sync Line Multiplexor (Micro Products)
		4463	Pearl Async/Sync Line Multiplexor (DG)
UPRINT	G	9999	Unicorn Printer
UPSC	G,3	0	Universal Power Supply Controller
UREAD	G	9999	Unicorn Card Reader
VID	G,1	4337	Video Interface Board (Micro Products)
WCS	G,1	8415	Writeable Control Store (S/200, S/230)
		8615	Writeable Control Store (S/130)
		8638	Writeable Control Store (S/250)

Table D-1. Mnemonic Codes and Descriptions (continued)

NOTES

G	General Device Format
C	Communications Device Format
D	Disk Device Format
T	Tape Device Format
1	Device(s) under this mnemonic or model number are not autosized.
2	ADEX uses only the removeable platter; therefore, disk storage capacity available to ADEX is only one-half of what is stated.
3	Model # of 0 implies processor feature is standard.
4	DTOS CAT support not available on this media.
5	Byte packed media; 512 words are required to read in one sector.

APPENDIX E
CORESIDENT INSTALLATION FLOWCHARTS

APPENDIX E
CORESIDENT INSTALLATION FLOWCHARTS

The following flowcharts are provided here to assist you with your CORESIDENT installation if it has not already been installed on your system. The CORESIDENT feature is explained in detail in Section 3. For details on preparing your disk and reserving space for the CORESIDENT feature, refer to the "CORESIDENT DIAGNOSTIC SYSTEM OPERATOR'S REFERENCE GUIDE" (DGC Part No. 014-001186).

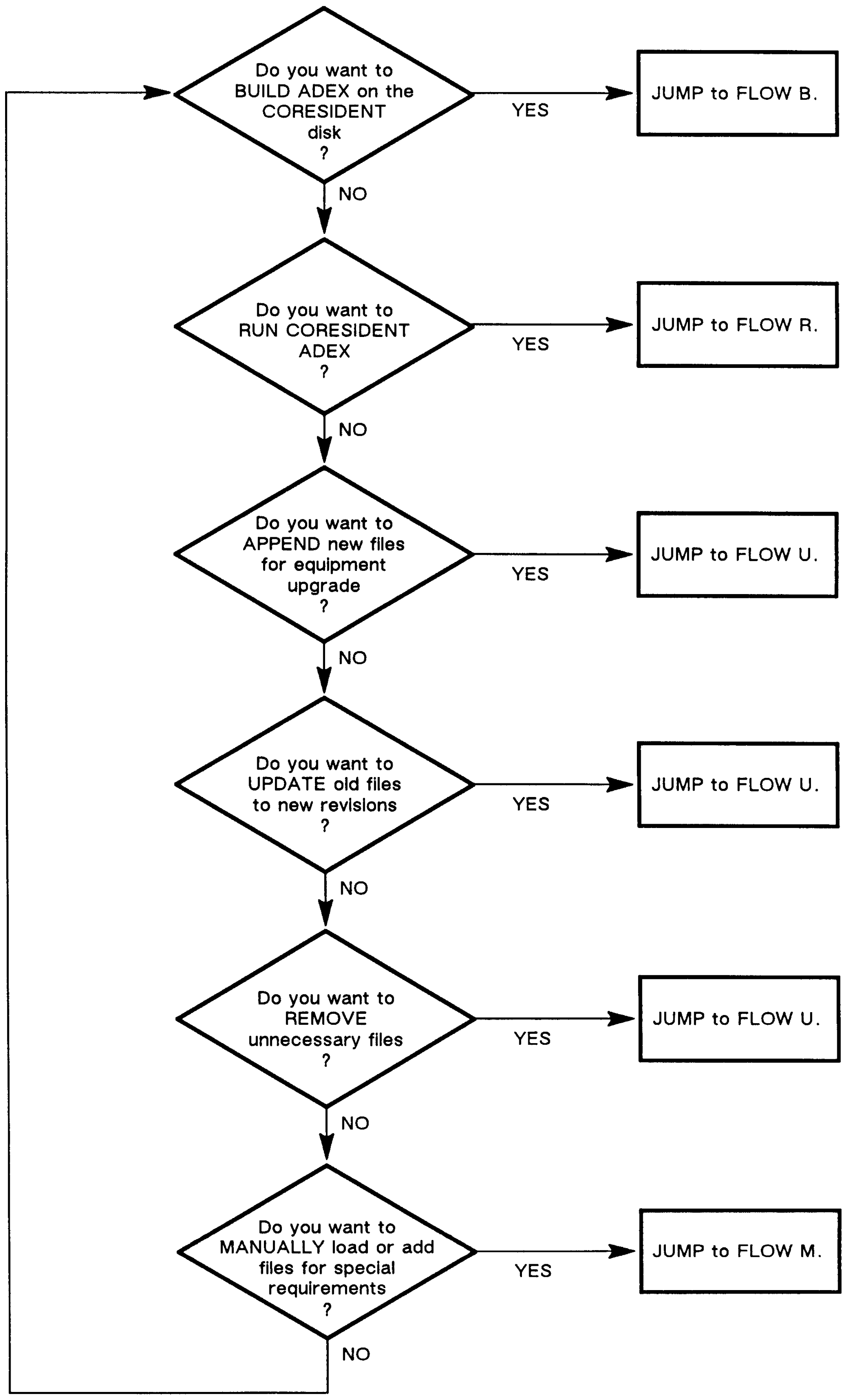
E.1 INSTALLATION NOTES

Please read the following notes before you start the CORESIDENT diagnostics installation.

1. The flow charts assume that you have some working knowledge of ADEX commands and utilities, although it is not a requirement to perform the installation.
2. Please note that actual ADEX commands are hi-lighted in BOLD after the ADEX CLI prompt: "ADEX-CLI>."

E.2 CORESIDENT MAIN MENU

Follow Figure E-1 to ensure that the system has been correctly prepared to load CORESIDENT ADEX. The directions in the main menu will instruct you to jump to FLOW B (Figure E-2), FLOW G (Figure E-3) for building from 96-TPI diskettes, FLOW R (Figure E-4), and FLOW U (Figure E-5) to build, run, and update or append files. Use FLOW M (Figure E-6) for special situations that require manually loading or adding of certain files.


Do you want to BUILD ADEX on the CORESIDENT disk ?
YES
JUMP to FLOW B.
NO
Do you want to RUN CORESIDENT ADEX ?
YES
JUMP to FLOW R.
NO
Do you want to APPEND new files for equipment upgrade ?
YES
JUMP to FLOW U.
NO
Do you want to UPDATE old files to new revisions ?
YES
JUMP to FLOW U.
NO
Do you want to REMOVE unnecessary files ?
YES
JUMP to FLOW U.
NO
Do you want to MANUALLY load or add files for special requirements ?
YES
JUMP to FLOW M.
NO

FS-12838

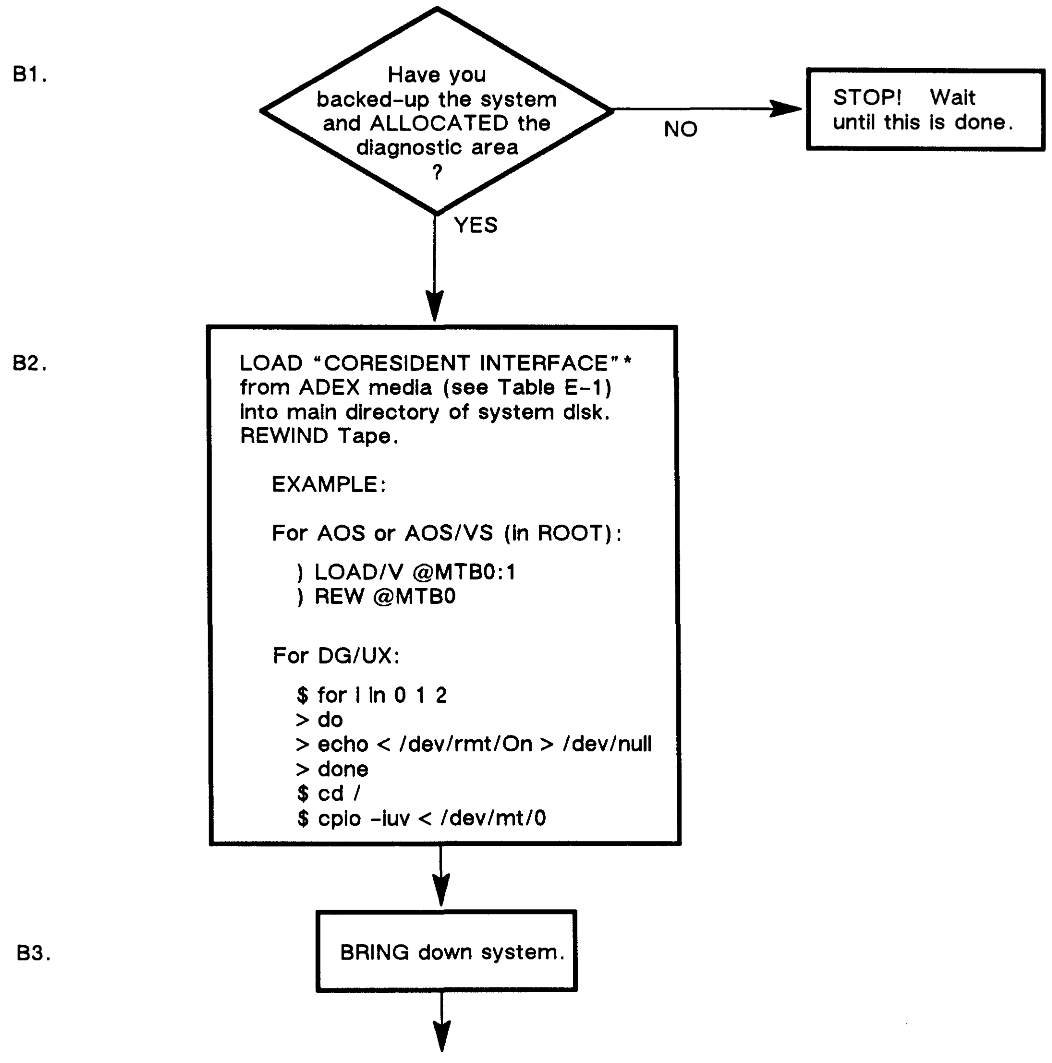
Figure E-1. CORESIDENT ADEX Main Menu

E.3 BUIDING CORESIDENT ADEX

Figure E-2 is a flow chart detailing the building instructions for CORESIDENT ADEX.

NOTE

DO NOT BRING DOWN SYSTEM YET!


B1.
Have you backed-up the system and ALLOCATED the diagnostic area ?
NO
STOP! Wait until this is done.
YES
B2.
B3.
BRING down system.

LOAD "CORESIDENT INTERFACE"* from ADEX media (see Table E-1) into main directory of system disk. REWIND Tape.

EXAMPLE:

For AOS or AOS/VS (in ROOT):

```
) LOAD/V @MTB0:1
) REW @MTB0
```

For DG/UX:

```
$ for i in 0 1 2
> do
> echo < /dev/rmt/On > /dev/null
> done
$ cd /
$ cpio -iuv < /dev/mt/0
```

* To save time on remote installations, have the system operator load the link file BEFORE the CORESIDENT installation (during the time period between scheduling and actual CORESIDENT installation date).

FS-12839

Figure E-2. Building CORESIDENT ADEX (Sheet 1 of 4)

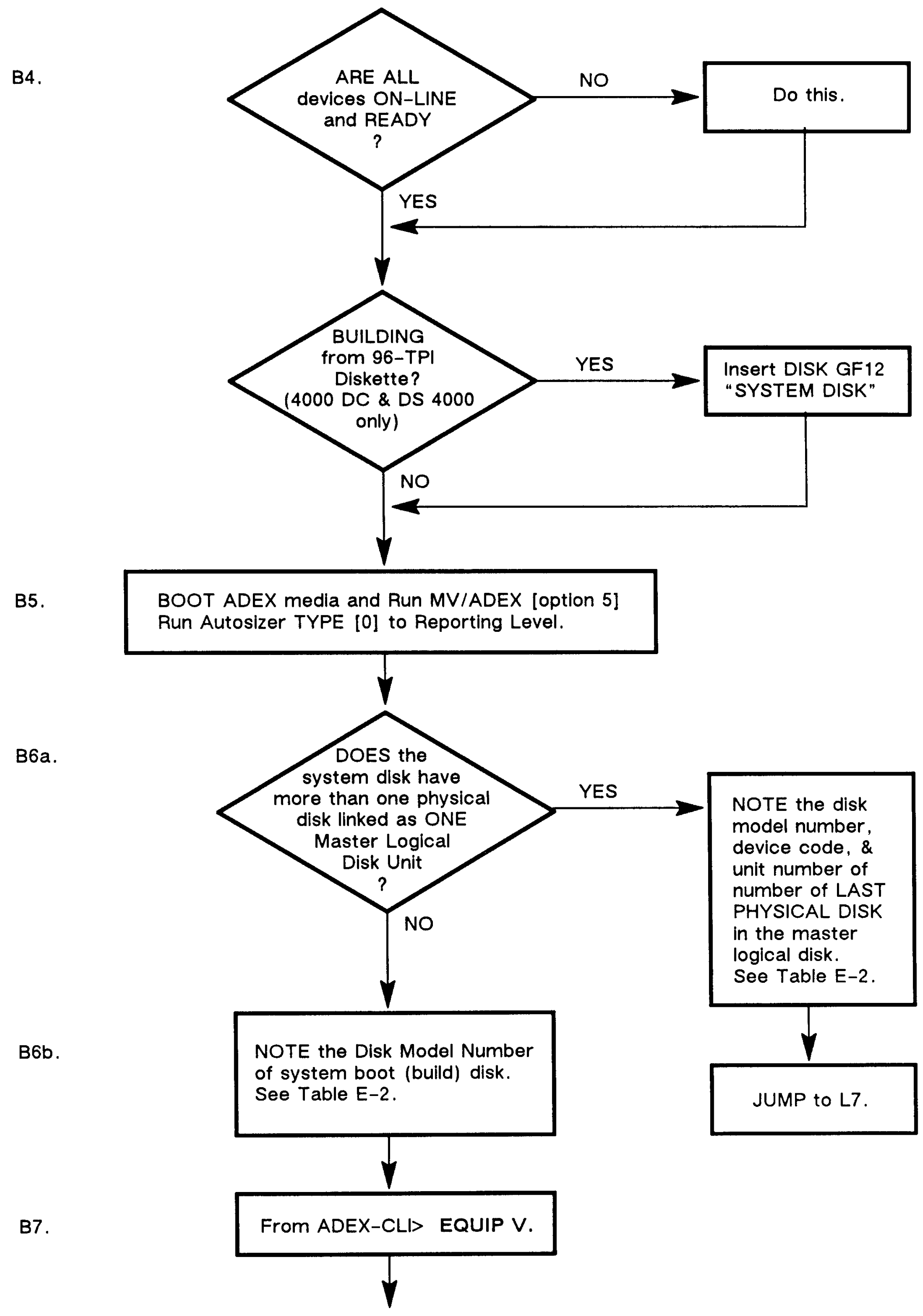
B4.

B5.

B6a.

B6b.

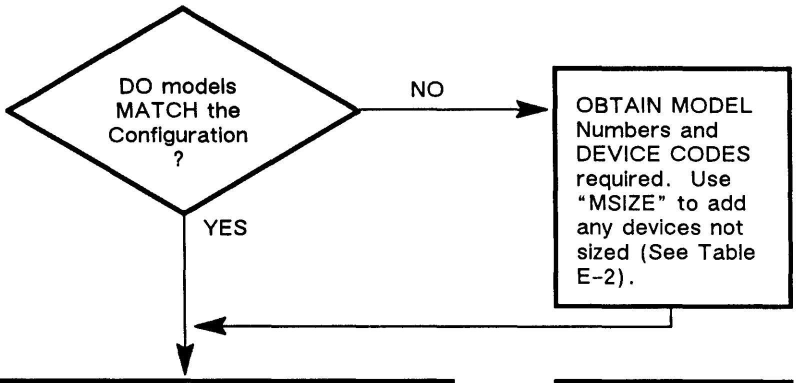
B7.


B4.
ARE ALL devices ON-LINE and READY ?
NO
Do this.
YES
BUILDING from 96-TPI Diskette? (4000 DC & DS 4000 only)
YES
Insert DISK GF12 "SYSTEM DISK"
NO
B5.
BOOT ADEX media and Run MV/ADEX [option 5] Run Autosizer TYPE [0] to Reporting Level.
B6a.
DOES the system disk have more than one physical disk linked as ONE Master Logical Disk Unit ?
YES
NOTE the disk model number, device code, & unit number of number of LAST PHYSICAL DISK in the master logical disk. See Table E-2.
NO
B6b.
NOTE the Disk Model Number of system boot (build) disk. See Table E-2.
JUMP to L7.
B7.
From ADEX-CLI> EQUIP V.

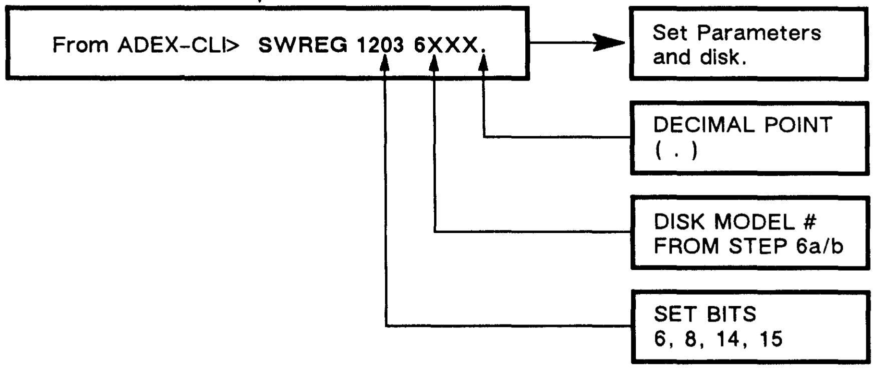
FS-12839

Figure E-2. Building CORESIDENT ADEX (Sheet 2 of 4)

B8.


DO models MATCH the Configuration ?
NO
OBTAIN MODEL Numbers and DEVICE CODES required. Use "MSIZE" to add any devices not sized (See Table E-2).
YES

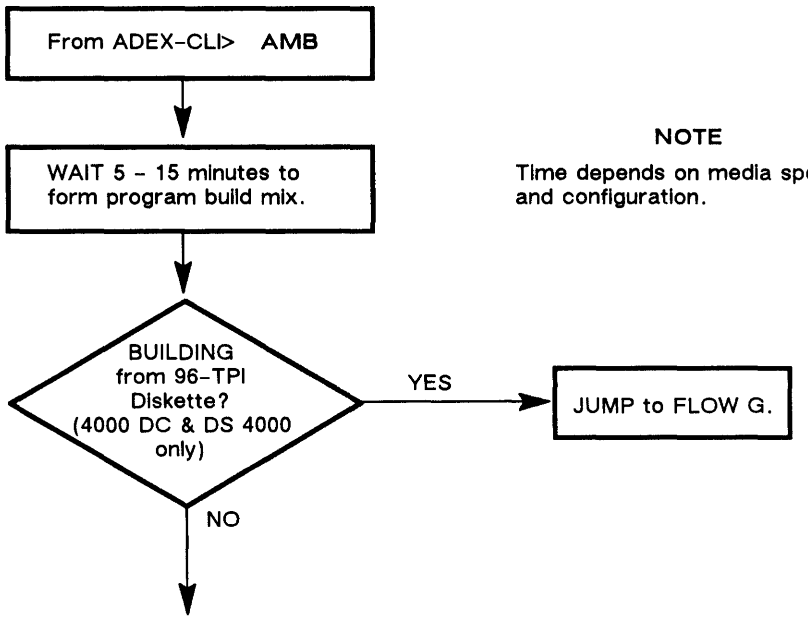
B9.



WARNING

Failure to input correct model number noted from step B6a. or B6b. will abort the disk build. Verify model number with system manager.

B10.


From ADEX-CLI> AMB
WAIT 5 - 15 minutes to form program build mix.
BUILDING from 96-TPI Diskette? (4000 DC & DS 4000 only)
YES
JUMP to FLOW G.
NO

NOTE

Time depends on media speed and configuration.

FS-12839

Figure E-2. Building CORESIDENT ADEX (Sheet 3 of 4)

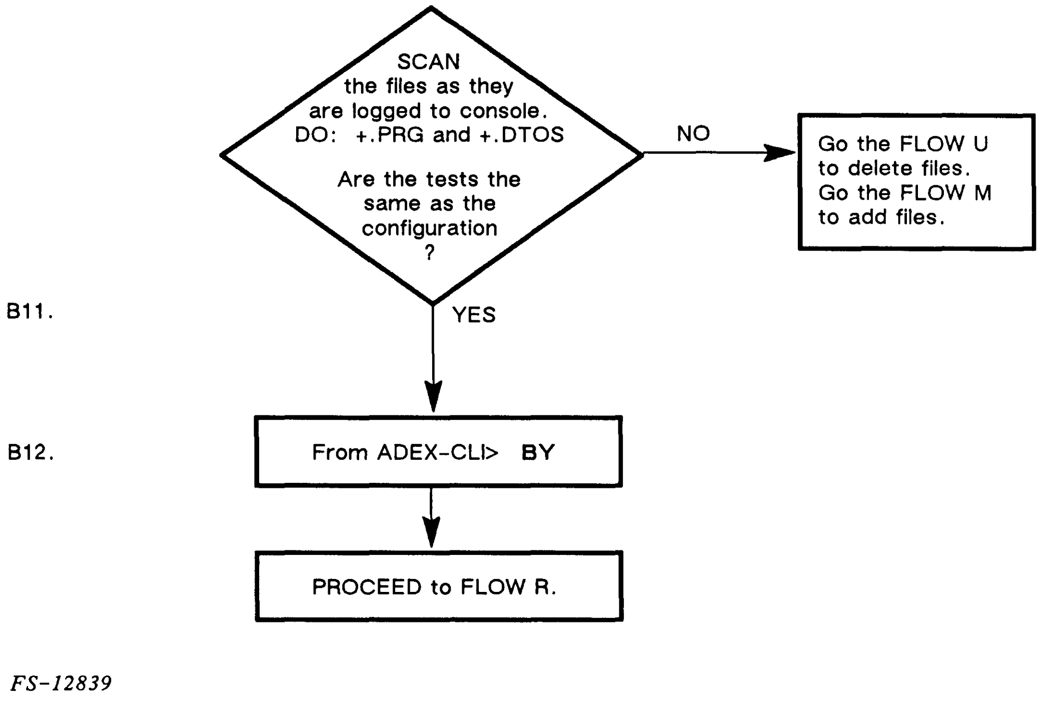

SCAN the files as they are logged to console. DO: +.PRG and +.DTOS
Are the tests the same as the configuration ?
NO
Go the FLOW U to delete files. Go the FLOW M to add files.
B11.
YES
B12.
From ADEX-CLI> **BY**
PROCEED to FLOW R.
FS-12839

Figure E-2. Building CORESIDENT ADEX (Sheet 4 of 4)

Table E-1. **ADEX Media with CORESIDENT**

MODEL NUMBER	DESCRIPTION	LINK PROGRAM (#) (3 FORMATS AVAILABLE)
30200-20H	MV ADEX REV 4.0	AOS/VS (1), RDOS (2), DG/UX (3)
30200-20C	MV ADEX REV 4.0	AOS/VS (1), RDOS (2), DG/UX (3)
30200-20G	MV ADEX REV 4.0	Last 3 Diskettes (as listed above)
30199-20M	N/E ADEX REV 5.0	AOS (1), RDOS (2)
30199-20H	N/E ADEX REV 5.0	AOS (1), RDOS (2)
30199-20C	N/E ADEX REV 5.0	AOS (1), RDOS (2)

H = 1600 BPI, C = Cartridge Tape, M = 800 BPI, G = 96 TPI Diskette

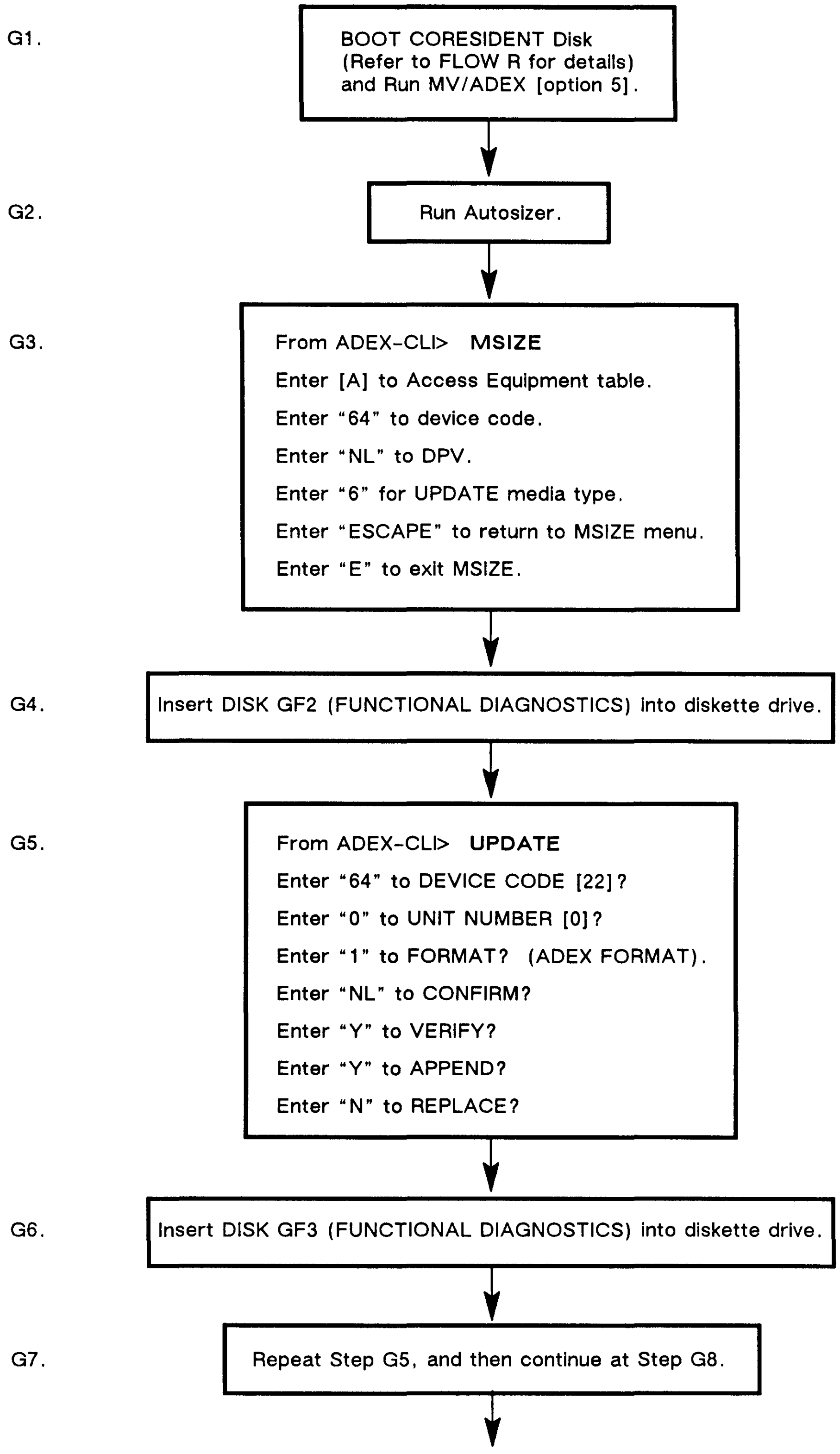
Table E-2 details the instructions for adding any devices not initially auto-sized by ADEX.

Table E-2. **MSIZE Instructions**

TERMINAL OUTPUT	YOUR RESPONSE IN BOLD	COMMENTS
ADEX-CLI>	[MSIZE]	Enter MSIZE Utility.
Specify operation (?,A,C,E,H,M,P) [A]:	[NEW LINE]	ACCESS equipment table.
Enter device code [--]: Device mnemonic [---]:	[Dev Code] [NEW LINE]	Enter new device code.
Modify or delete this entry (M,D) [M]?	[NEW LINE]	To MODIFY this entry.
Unit #'s to access [0]:	[Unit #]	Enter unit number.
Model # [XXXX]:	[Model #]	Enter model number.
Media type (?, 0-9) [2]? ?	[?]	Determine media type.
Enter device code [--]:		Enter additional device code(s) and repeat sequence, or [ESCAPE] to return to MSIZE menu.
Specify operation (?,A,C,E,H,M,P) [A]:	[E]	To exit MSIZE.
Place new equipment table into memory ..?	[Y]	Save new equipment table.
ADEX-CLI>		Proceed to next step.

NOTE

Use of MSIZE utility will reset the SWREG to 0s, so SWREG must be set AFTER use of MSIZE.



G1. BOOT CORESIDENT Disk (Refer to FLOW R for details) and Run MV/ADEX [option 5].

G2. Run Autosizer.

G3. From ADEX-CLI> **MSIZE**
Enter [A] to Access Equipment table.
Enter "64" to device code.
Enter "NL" to DPV.
Enter "6" for UPDATE media type.
Enter "ESCAPE" to return to MSIZE menu.
Enter "E" to exit MSIZE.

G4. Insert DISK GF2 (FUNCTIONAL DIAGNOSTICS) into diskette drive.

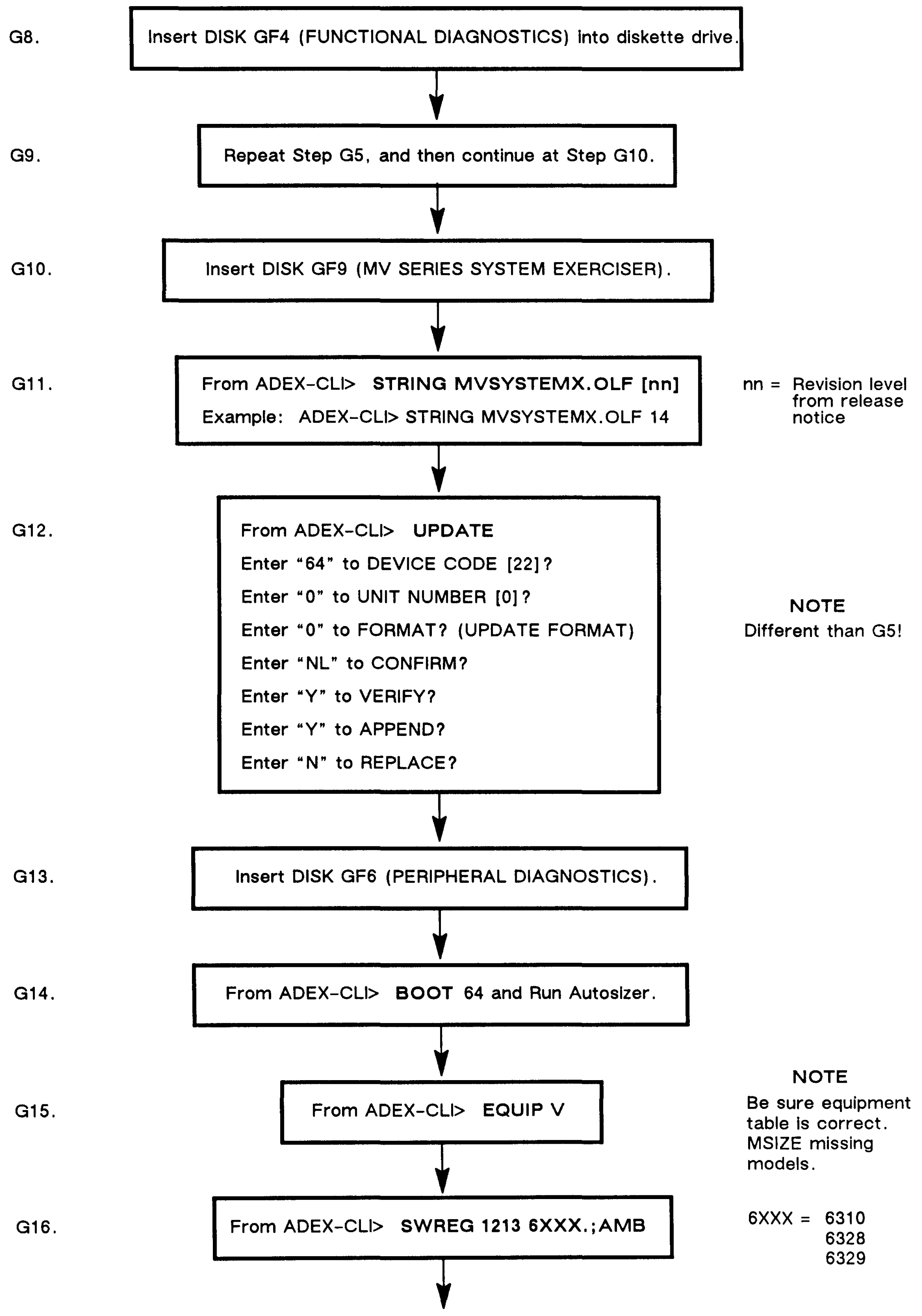
G5. From ADEX-CLI> **UPDATE**
Enter "64" to DEVICE CODE [22]?
Enter "0" to UNIT NUMBER [0]?
Enter "1" to FORMAT? (ADEX FORMAT).
Enter "NL" to CONFIRM?
Enter "Y" to VERIFY?
Enter "Y" to APPEND?
Enter "N" to REPLACE?

G6. Insert DISK GF3 (FUNCTIONAL DIAGNOSTICS) into diskette drive.

G7. Repeat Step G5, and then continue at Step G8.

FS-12840

Figure E-3. Building With 96-TPI Diskettes (DS 4000 & 4000 DC)
(Sheet 1 of 3)



G8. Insert DISK GF4 (FUNCTIONAL DIAGNOSTICS) into diskette drive.

G9. Repeat Step G5, and then continue at Step G10.

G10. Insert DISK GF9 (MV SERIES SYSTEM EXERCISER).

G11. From ADEX-CLI> **STRING MVSYSTEMX.OLF [nn]**

Example: ADEX-CLI> STRING MVSYSTEMX.OLF 14

nn = Revision level from release notice

G12. From ADEX-CLI> **UPDATE**

Enter "64" to DEVICE CODE [22]?

Enter "0" to UNIT NUMBER [0]?

Enter "0" to FORMAT? (UPDATE FORMAT)

Enter "NL" to CONFIRM?

Enter "Y" to VERIFY?

Enter "Y" to APPEND?

Enter "N" to REPLACE?

NOTE

Different than G5!

G13. Insert DISK GF6 (PERIPHERAL DIAGNOSTICS).

G14. From ADEX-CLI> **BOOT** 64 and Run Autosizer.

G15. From ADEX-CLI> **EQUIP V**

NOTE

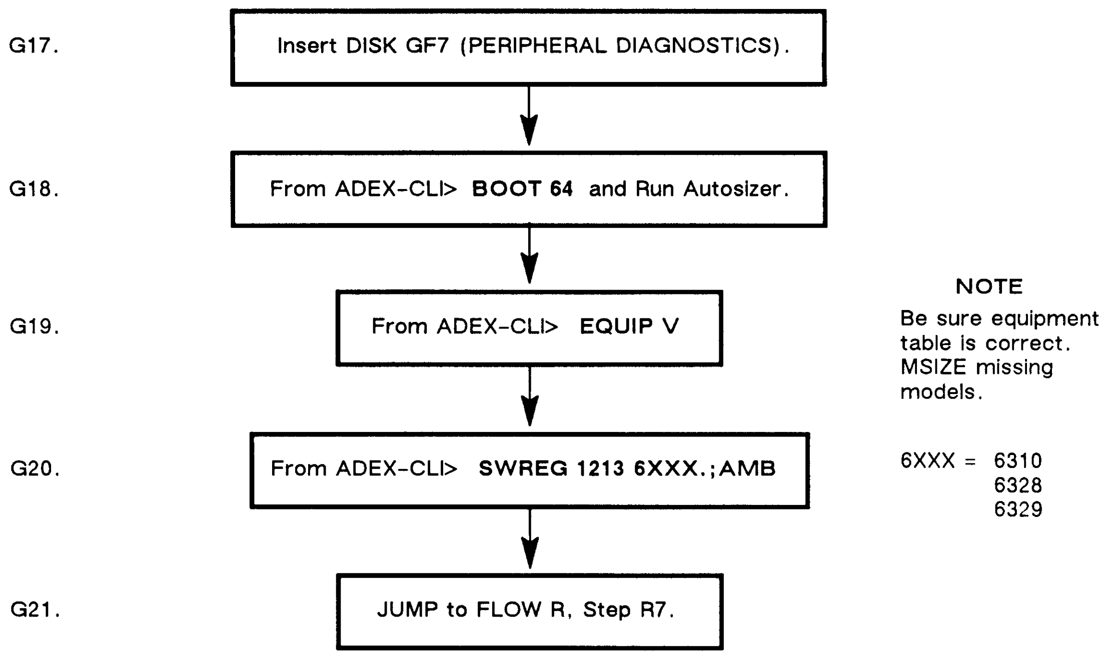
Be sure equipment table is correct. MSIZE missing models.

G16. From ADEX-CLI> **SWREG 1213 6XXX.;AMB**

6XXX = 6310
6328
6329

FS-12840

Figure E-3. Building With 96-TPI Diskettes (DS 4000 & 4000 DC) (Sheet 2 of 3)


G17. Insert DISK GF7 (PERIPHERAL DIAGNOSTICS).

G18. From ADEX-CLI> **BOOT 64** and Run Autosizer.

G19. From ADEX-CLI> **EQUIP V**

G20. From ADEX-CLI> **SWREG 1213 6XXX.;AMB**

G21. JUMP to FLOW R, Step R7.

NOTE
Be sure equipment table is correct. MSIZE missing models.

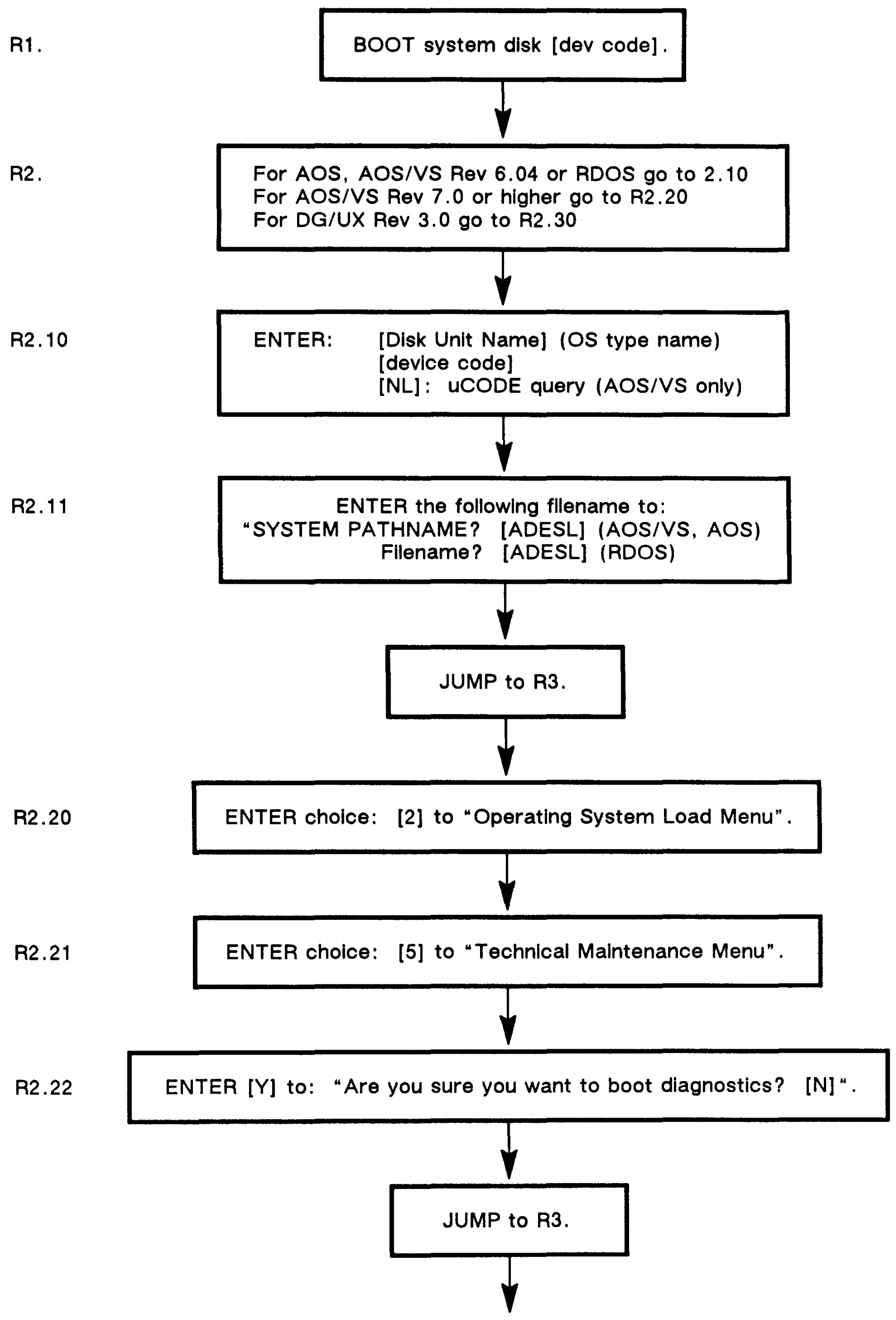
```
6XXX =  6310
        6328
        6329
```

FS-12840

Figure E-3. Building With 96-TPI Diskettes (DS 4000 & 4000 DC) (Sheet 3 of 3)

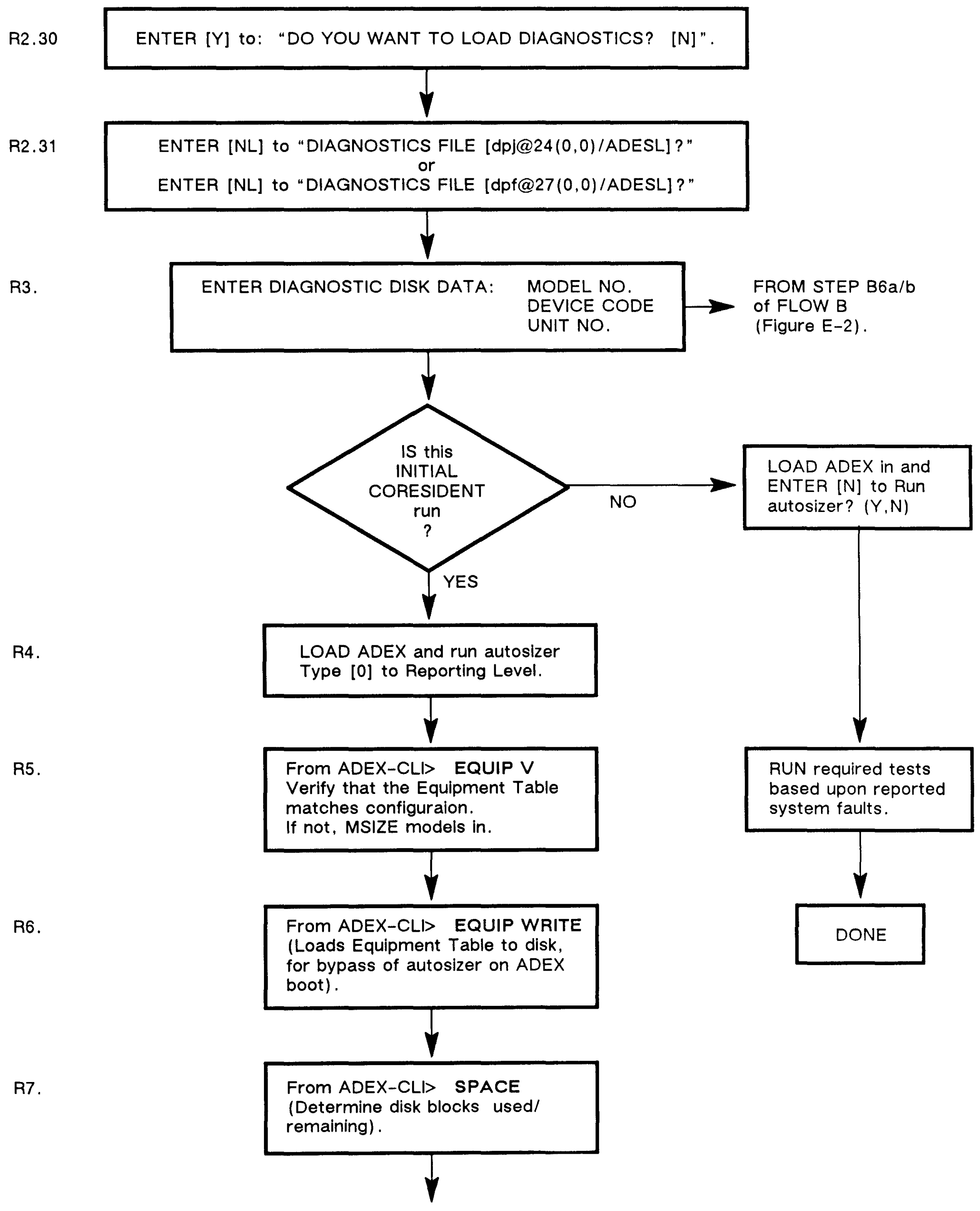
E.4 RUNNING CORESIDENT ADEX

Figure E–4 details the procedure for running the CORESIDENT ADEX.


R1.
BOOT system disk [dev code].
R2.
For AOS, AOS/VS Rev 6.04 or RDOS go to 2.10
For AOS/VS Rev 7.0 or higher go to R2.20
For DG/UX Rev 3.0 go to R2.30
R2.10
ENTER: [Disk Unit Name] (OS type name)
[device code]
[NL]: uCODE query (AOS/VS only)
R2.11
ENTER the following filename to:
"SYSTEM PATHNAME? [ADESL] (AOS/VS, AOS)
Filename? [ADESL] (RDOS)
JUMP to R3.
R2.20
ENTER choice: [2] to "Operating System Load Menu".
R2.21
ENTER choice: [5] to "Technical Maintenance Menu".
R2.22
ENTER [Y] to: "Are you sure you want to boot diagnostics? [N]".
JUMP to R3.

FS-12841

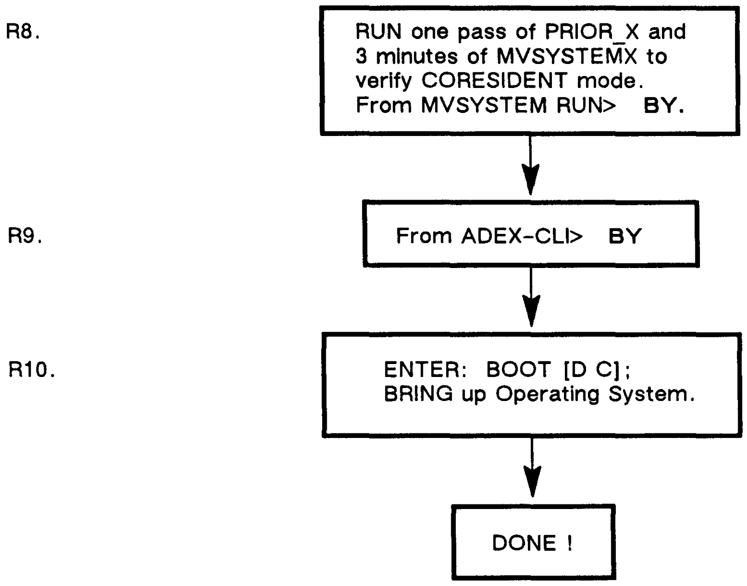
Figure E–4. Running CORESIDENT ADEX (Sheet 1 of 3)



R2.30
ENTER [Y] to: "DO YOU WANT TO LOAD DIAGNOSTICS? [N]".
R2.31
ENTER [NL] to "DIAGNOSTICS FILE [dpj@24(0,0)/ADESL]?"
or
ENTER [NL] to "DIAGNOSTICS FILE [dpf@27(0,0)/ADESL]?"
R3.
ENTER DIAGNOSTIC DISK DATA: MODEL NO. DEVICE CODE UNIT NO.
FROM STEP B6a/b of FLOW B (Figure E-2).
IS this INITIAL CORESIDENT run ?
NO
LOAD ADEX in and ENTER [N] to Run autosizer? (Y,N)
YES
R4.
LOAD ADEX and run autosizer Type [0] to Reporting Level.
R5.
From ADEX-CLI> EQUIP V Verify that the Equipment Table matches configuraion. If not, MSIZE models in.
RUN required tests based upon reported system faults.
DONE
R6.
From ADEX-CLI> EQUIP WRITE (Loads Equipment Table to disk, for bypass of autosizer on ADEX boot).
R7.
From ADEX-CLI> SPACE (Determine disk blocks used/ remaining).

FS-12841

Figure E-4. Running CORESIDENT ADEX (Sheet 2 of 3)

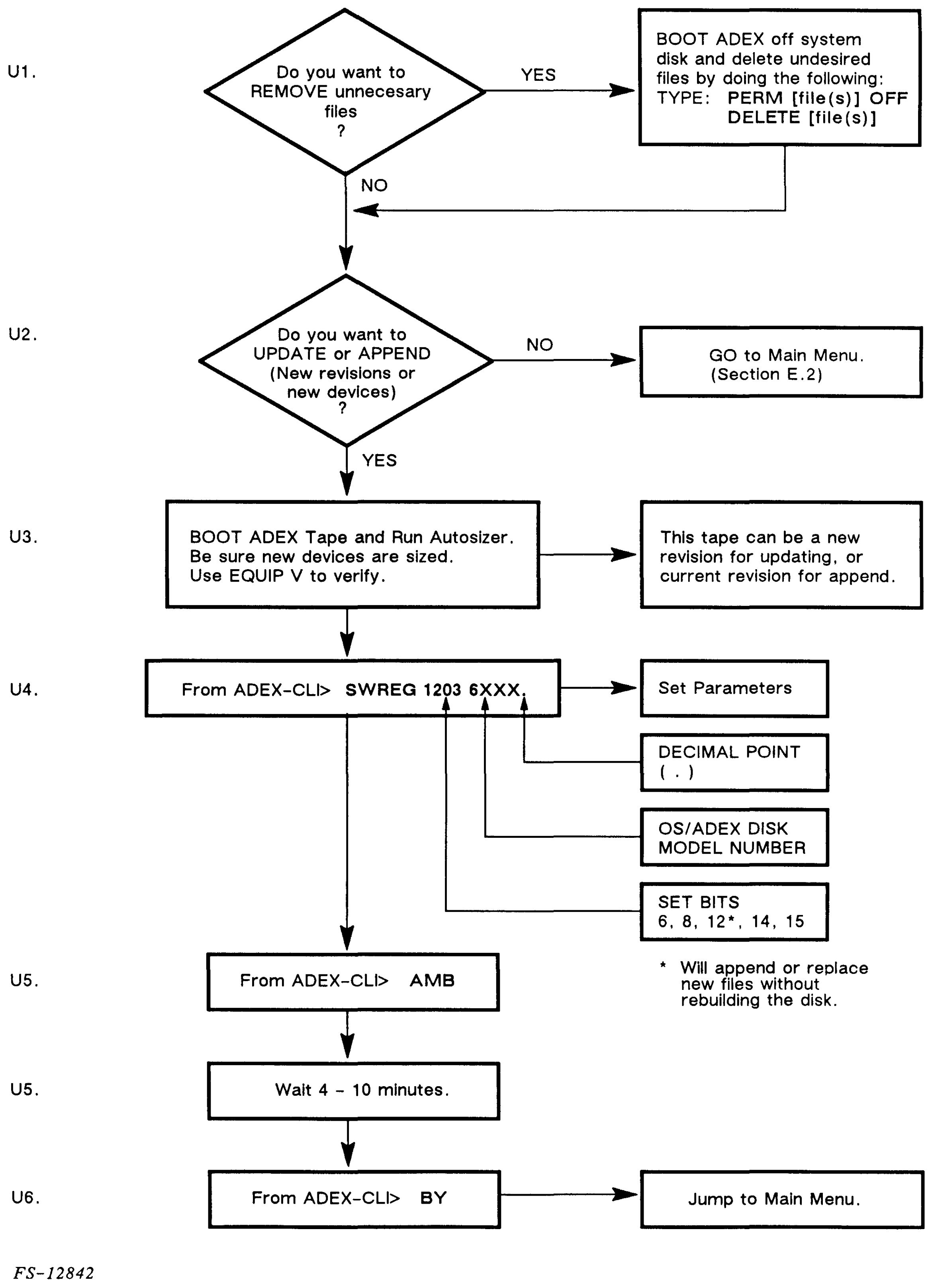

R8.
RUN one pass of PRIOR_X and 3 minutes of MVSYSTEMX to verify CORESIDENT mode. From MVSYSTEM RUN> BY.
R9.
From ADEX-CLI> BY
R10.
ENTER: BOOT [D C]; BRING up Operating System.
DONE !

FS-12841

Figure E-4. Running CORESIDENT ADEX (Sheet 3 of 3)

E.5 UPDATING OR APPENDING CORESIDENT ADEX

Figure E–5 gives detailed instructions necessary for updating or appending CORESIDENT ADEX.



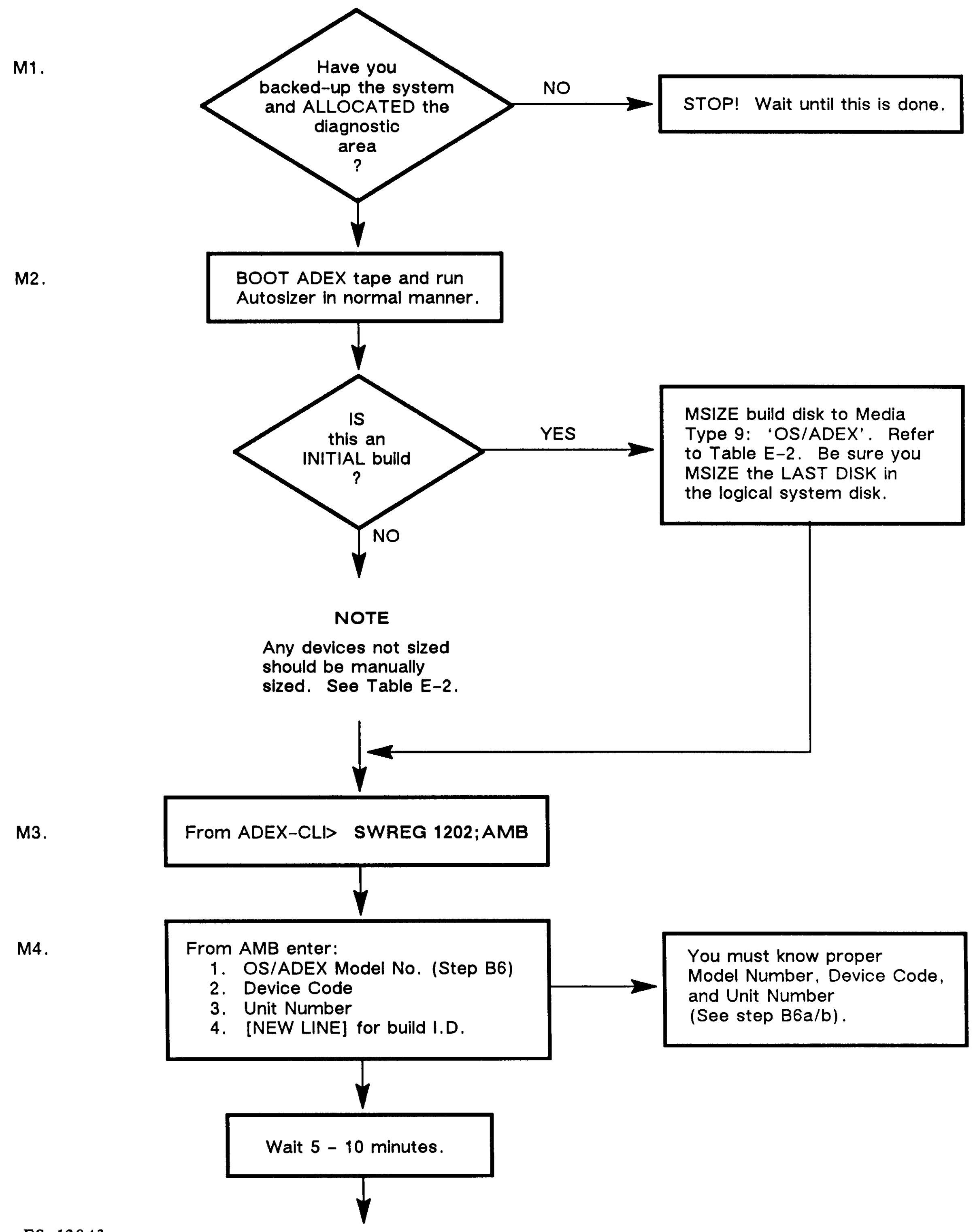
U1.
Do you want to REMOVE unnecesary files ?
YES
BOOT ADEX off system disk and delete undesired files by doing the following:
TYPE: PERM [file(s)] OFF
DELETE [file(s)]
NO
U2.
Do you want to UPDATE or APPEND (New revisions or new devices) ?
NO
GO to Main Menu. (Section E.2)
YES
U3.
BOOT ADEX Tape and Run Autosizer. Be sure new devices are sized. Use EQUIP V to verify.
This tape can be a new revision for updating, or current revision for append.
U4.
From ADEX-CLI> SWREG 1203 6XXX.
Set Parameters
DECIMAL POINT (.)
OS/ADEX DISK MODEL NUMBER
SET BITS 6, 8, 12*, 14, 15
* Will append or replace new files without rebuilding the disk.
U5.
From ADEX-CLI> AMB
U5.
Wait 4 - 10 minutes.
U6.
From ADEX-CLI> BY
Jump to Main Menu.

FS-12842

Figure E–5. Updating or Appending CORESIDENT ADEX

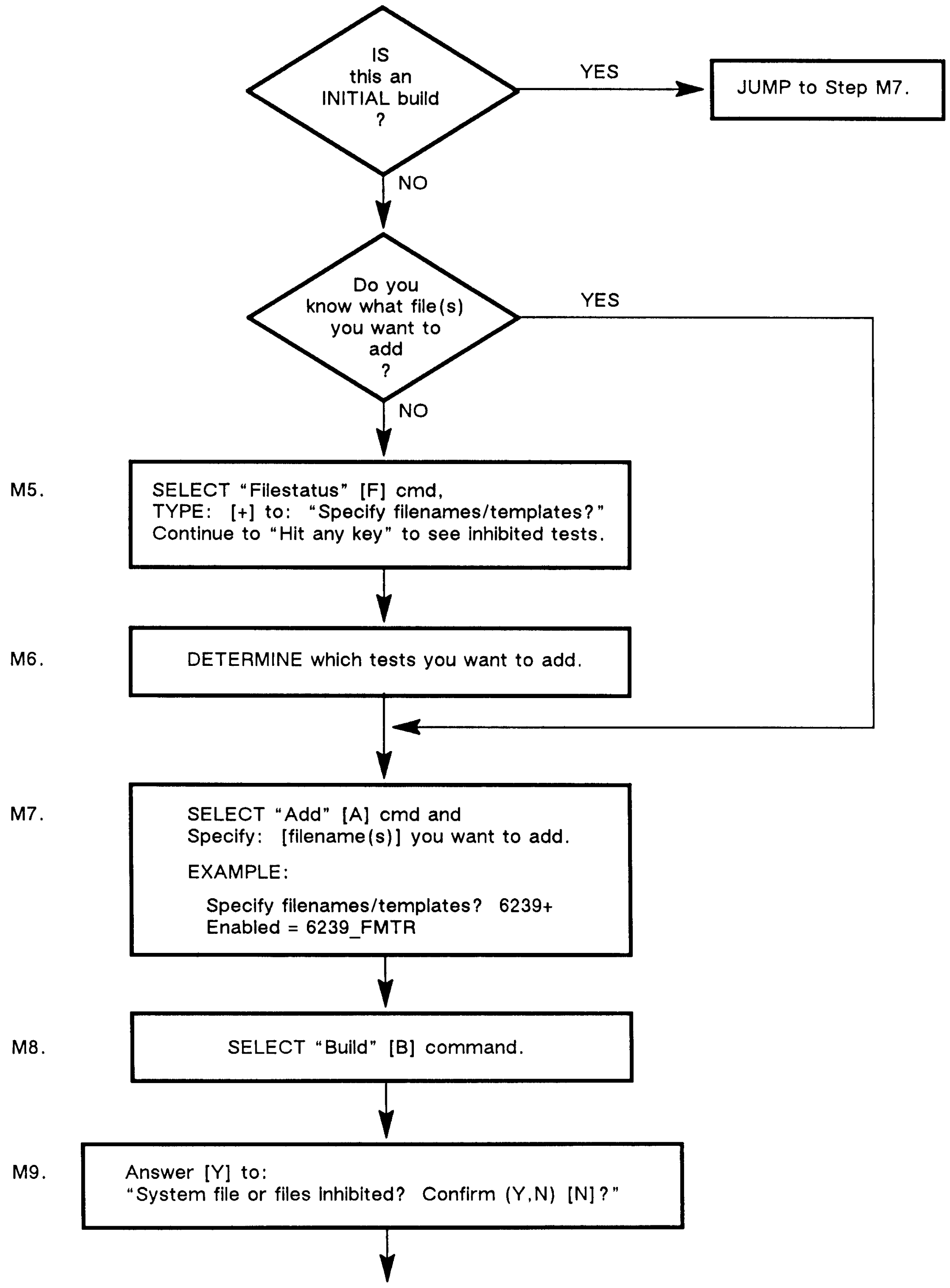
E.6 MANUALLY BUILDING OR APPENDING CORESIDENT ADEX

Figure E-6 gives detailed instructions necessary for manually doing an initial build or appending files for special situations.


M1.
Have you backed-up the system and ALLOCATED the diagnostic area ?
NO
STOP! Wait until this is done.
M2.
BOOT ADEX tape and run Autosizer in normal manner.
IS this an INITIAL build ?
YES
MSIZE build disk to Media Type 9: 'OS/ADEX'. Refer to Table E-2. Be sure you MSIZE the LAST DISK in the logical system disk.
NO
NOTE
Any devices not sized should be manually sized. See Table E-2.
M3.
From ADEX-CLI> SWREG 1202;AMB
M4.
From AMB enter:
1. OS/ADEX Model No. (Step B6)
2. Device Code
3. Unit Number
4. [NEW LINE] for build I.D.
You must know proper Model Number, Device Code, and Unit Number (See step B6a/b).
Wait 5 - 10 minutes.

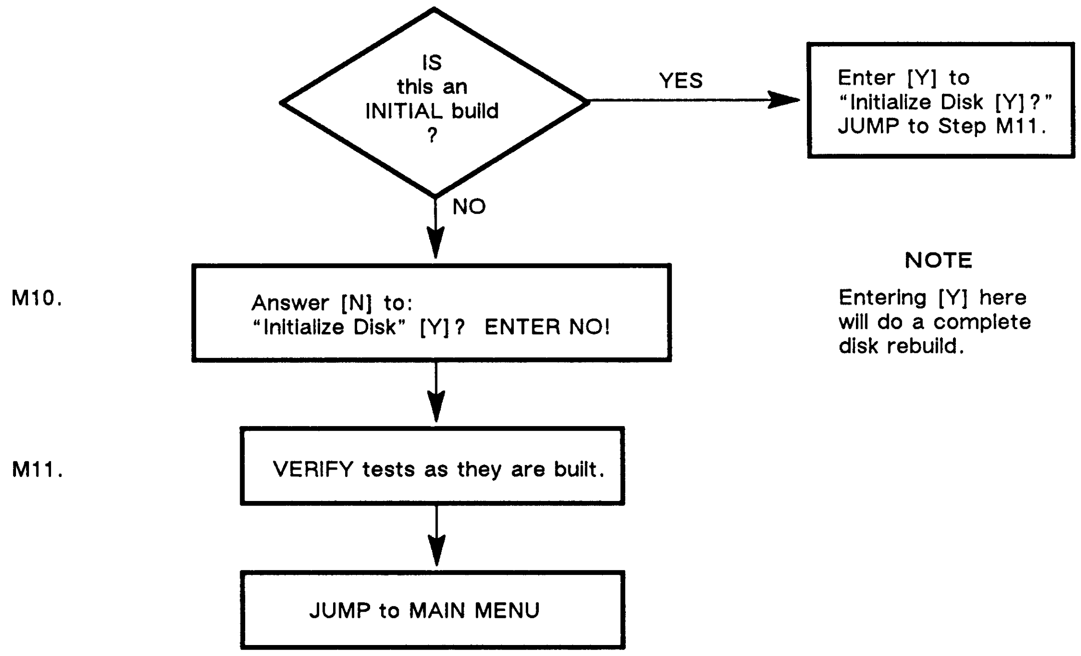
FS-12843

Figure E-6. Manually Building or Adding Files for Special Requirements (Sheet 1 of 3)


IS this an INITIAL build ?
YES
JUMP to Step M7.
NO
Do you know what file(s) you want to add ?
YES
NO
M5.
SELECT "Filestatus" [F] cmd,
TYPE: [+] to: "Specify filenames/templates?"
Continue to "Hit any key" to see inhibited tests.
M6.
DETERMINE which tests you want to add.
M7.
SELECT "Add" [A] cmd and
Specify: [filename(s)] you want to add.
EXAMPLE:
Specify filenames/templates? 6239+
Enabled = 6239_FMTR
M8.
SELECT "Build" [B] command.
M9.
Answer [Y] to:
"System file or files inhibited? Confirm (Y,N) [N]?"

FS-12843

Figure E-6. Manually Building or Adding Files for Special Requirements (Sheet 2 of 3)



IS this an INITIAL build ?
YES
Enter [Y] to "Initialize Disk [Y]?" JUMP to Step M11.
NO
NOTE
Entering [Y] here will do a complete disk rebuild.
M10.
Answer [N] to: "Initialize Disk" [Y]? ENTER NO!
M11.
VERIFY tests as they are built.
JUMP to MAIN MENU

FS-12843

Figure E-6. Manually Building or Adding Files for Special Requirements (Sheet 3 of 3)

APPENDIX F
CONDITIONAL CLI

APPENDIX F
CONDITIONAL CLI

You can create powerful scripts through the use of conditional CLI commands. These commands are executed only if the condition of the software control flag is met. ADEX includes six software control flags:

R0
R1
R2
R3
WARNING ERROR

These flags can all be accessed by user programs (so the flags can be set from within a program). The ERROR flag is set automatically whenever a diagnostic program finds an error. The other flags can be set with the **SET** command. In addition, the WARNING flag is set whenever the CLI encounters an error in a command line. You can clear any flag with the **CLEAR** command.

To check the status of any flag, use the **TEST** command. Depending on the result of the test, the CLI then continues at a particular node, or point, in the script. You can define these nodes with the **LABEL** command.

Figure F-1 shows a script that contains conditional CLI statements. The purpose of this script is to run a test of the ECLIPSE® Standard Instruction Set and test for an error. The conditional statements in the script will ensure that if there is an error, the R0 flag is set and a message is printed out. If there is no error, the R0 flag is cleared, and a different message is printed.

```
ADEX-CLI> TYpe TEST.SRCE

   WRITE **Testing ECLIPSE Standard Instruction Set**
   X EASIS_X1
   X EASIS_X2
   X EASIS_X3
   X EASIS_X4
   TEST ERROR YES
   CLEAR R0; WRITE **TESTS PASSED**; TEST R0 EXIT EXIT
   LABEL YES; SET R0; WRITE **TESTS FAILED**
   LABEL EXIT
```

FS-12844

Figure F-1. A Sample of a Conditional CLI Script

In the script shown in Figure F–1, the first line prints out on your console. Then, the four ECLIPSE® Standard Instruction Sets are executed sequentially. Next, the ERROR flag is tested to see whether it is set to true or false. If the flag is true, then the R0 flag is set and the message TESTS FAILED is printed out. If it is false, the R0 flag is cleared and the message TESTS PASSED is printed out. The script ends at EXIT whether this flag is set at true or false.

APPENDIX G
CLI MACROS

APPENDIX G
CLI MACROS

Macros provide a short-cut in executing sequences of diagnostic programs, CLI commands, or both. Instead of typing in command lines one by one, enter them all in a macro file and then run the script. Macros are useful either when executing the same sequence of commands frequently or when there are too many commands to fit on a multiple command line. If writing a macro for such a lengthy command sequences, there is no waiting for one test to finish before typing in the execution line for the next command. Examples for macros are provided in the text.

WARNING

ADEX CLI commands are valid in script mode, but Edit commands are not.

To write a macro, create a text file known as a source file. This file contains a series of CLI commands. To execute the commands in the macro file, just type in the name of the file.

NOTE

CLI macros can also be written under AOS and can be included in the ADEX media by inserting the name of the CLI source file in the input list.

G.1 NESTING MACROS

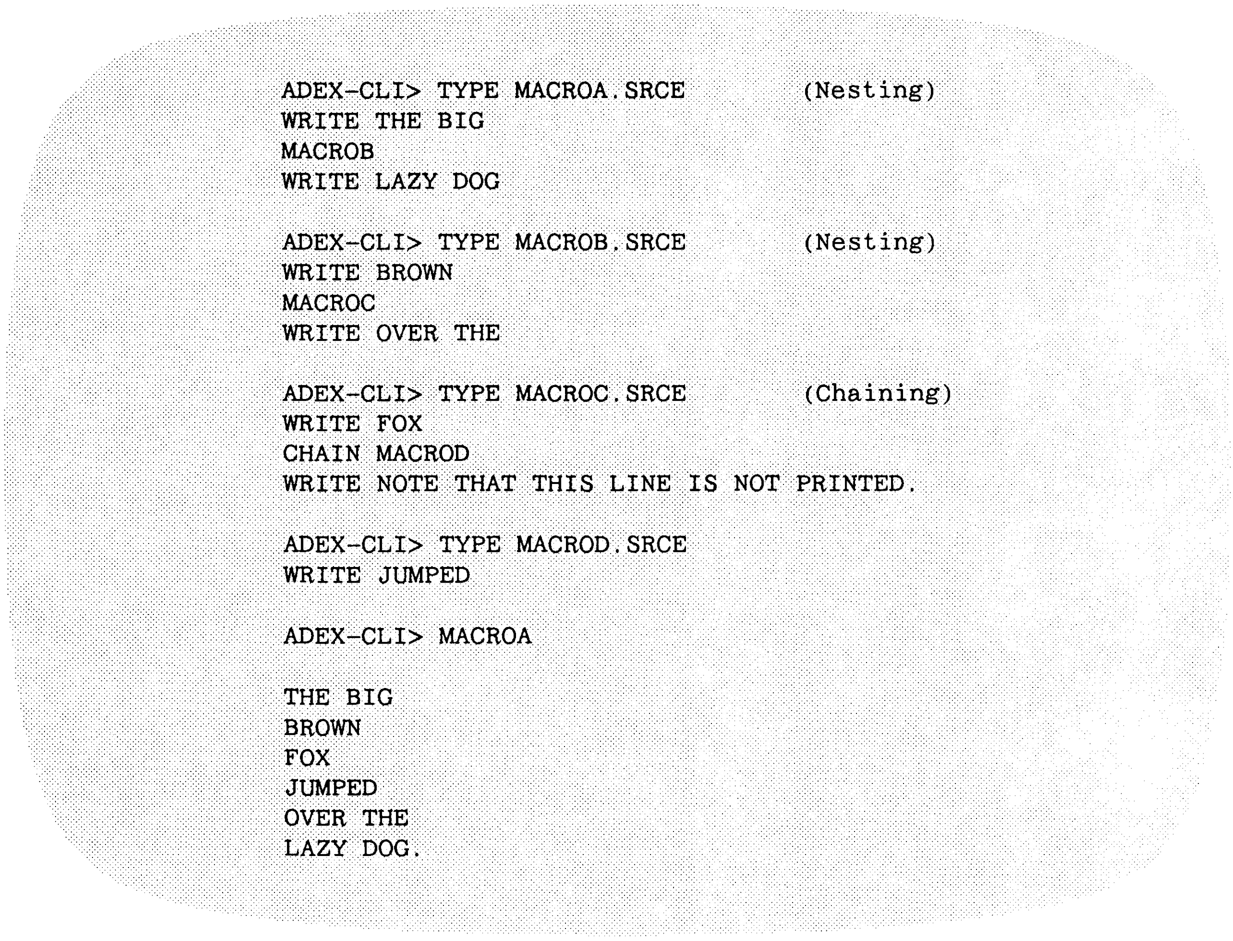
Long and repetitious macros can be avoided by having one macro call another. For example, one macro may be XYZ that performs a common operation which is a part of a number of different procedure. The macro for the larger procedure would include a command line with the name XYZ on it. When the CLI came to this line, it would execute the contents of macro file XYZ.

It is not necessary to use the .SRCE extension unless the name of the macro is also the name of a CLI command or its abbreviation. For example, if there was a macro called TYPE, TYPE.SRCE would have to be used so that the macro is executed rather than the CLI **TYPE** command.

The process of referring to other macros inside a macro file is known as nesting. When a nested macro finishes, the CLI returns to the next line in the original macro. There can be up to three levels of nesting.

G.2 CHAINING MACROS

In addition to nesting macros, you can link them with **CHAIN** command. To chain to another macro, when that macro finishes the prompt does not return to the macro that contained the **CHAIN**, but to the macro or CLI level above it. This gives you the choice of executing commands in a pattern that differs from nesting. An example of nesting and chaining macros is given in Figure G–1.



```
ADEX-CLI> TYPE MACROA.SRCE        (Nesting)
WRITE THE BIG
MACROB
WRITE LAZY DOG

ADEX-CLI> TYPE MACROB.SRCE        (Nesting)
WRITE BROWN
MACROC
WRITE OVER THE

ADEX-CLI> TYPE MACROC.SRCE        (Chaining)
WRITE FOX
CHAIN MACROD
WRITE NOTE THAT THIS LINE IS NOT PRINTED.

ADEX-CLI> TYPE MACROD.SRCE
WRITE JUMPED

ADEX-CLI> MACROA

THE BIG
BROWN
FOX
JUMPED
OVER THE
LAZY DOG.
```

FS-12845

Figure G–1. Nesting and Chaining Macros

G.3 MACRO ARGUMENTS

All arguments supplied on the command line are passed to the macro. The user can supply arguments through the ADEX Dummy Argument Facility. Dummy arguments are enclosed in percent signs and have effect only in the body of the macro. When ADEX invokes a macro, it generates and executes a temporary SRCE? file, with all the dummy arguments replaced with the requested macro arguments.

ADEX supports three dummy argument formats:

1. %N% – Insert arguement N.
2. %N–% – Insert argument N through last.
3. %N–M% – Insert argument N through M.

Argument 0 is the name of the macro. If a reference is made to a non–existent argument, ADEX ignores it. If an invalid format is used, the requested macro will not be run, and the error INVALID DUMMY ARGUMENT FORMAT IN MACRO results. Note that macro argument numbers are parsed in decimal. Figure G–2 shows an example.

```
ADEX-CLI> TYPE SAMPLE1.SRCE
WR HELLO<54>%1% %2%
WR I SEE YOU ARE %3-4% OLD, AND YOU ARE A %5-%
WR GOOD FOR YOU!! %8-%

ADEX-CLI>SAMPLE1 JOHN SMITH, 29 YEARS, SENIOR ACCOUNT EXECUTIVE
HELLO, JOHN SMITH
I SEE YOU ARE 29 YEARS OLD, AND YOU ARE A SENIOR ACCOUNT EXECUTIVE
GOOD FOR YOU!!

ADEX-CLI>
```

FS-12846

Figure G–2. Macro Argument

G.4 CREATING MACROS

Macros can be created in two ways:

1. By using the ADEX Script Builder.

2. By using the Edit utility.

G.4.1 ADEX Script Builder

The ADEX Script Builder (SCRBLD) is used to generate a list of programs that run on the current CPU, test equipment that is listed in the SET, and can be executed without asking any questions. On a tape, these programs are in the form of a memory–resident script that is executed and lost when the script finishes. On a disk, however, there is a choice. The programs can be in the form of a memory–resident script, or they can be in the form of a source file that contains a series of execute commands which can be saved on the runtime disk for later use.

To create either script, SCRBLD uses information from the System Equipment Table, from a cross–reference table which associates test programs with different devices, and from the directory structure which lists all the test programs on the runtime media.

G.4.2 The Edit Utility

The Edit utility lets you create and edit script files (in additon to other file types) on disk. With this utility, you can insert, delete, modify, and display text. You can run a one–page macro right from the Edit utility. Finally, you have the option of saving files on the disk runtime media.

With Edit, very powerful scripts can be created. For example, it is possible to create scripts which include conditional statements. Such statements let you control the execution of programs depending on certain conditions.

G.4.2.1 Using Edit to Create and Run a Script – Use Edit to create a script that will change the system default pass count value to 2, set the error class to warning, and display the contents of the System Equipment Table, including parameters. In addition, a message appears when these changes have been made.

Name the file TRIAL. To use the Edit utility to create this script, type:

```
ADEX-CLI> EDit TRIAL.SRCE
```

The system responds with the question:

```
CREATE NEW FILE (Y,N) [N]?
```

Type in **Y**, followed by <NL>. The script can now be entered. First, enter the insert mode in order to type in the appropriate commands. To do this, type:

```
EDIT-CLI> INSERT
```

The insert mode is now activated. The prompt has changes to an asterisk plus a line number as follows:

```
*1 DEFault PAsmax 2
*2 EClass Warning
*3 EQuipment Read
*4 Write CHANGES ARE INCORPORATED.
*5 $
```

NOTE

On line five, the user presses the ESC key. The system responds by printing out the dollar sign.

The EDIT-CLI prompt is back. To save the new file, and return to ADEX-CLI, type **BYE**. To execute this file from EDIT-CLI, type:

```
EDIT-CLI> EXECUTE
```

The execution must be confirmed. The file is not saved if executed from EDIT-CLI.

To execute a file from ADEX-CLI, type in the name of the file.

G.4.2.2 Writing Scripts – When writing scripts, the **READ** and **WRITE** commands can be incorporated. To insert a pause in a macro to allow the operator to do a certain task, use **READ** with the appropriate message as the arguement. **WRITE** (see the previous example) prints a message in the macro and is used to let the operator know that a particular event has taken place.

INDEX

134-757-01

moisten & seal

Data General Service, Inc.

FIELD ENGINEERING TECHNICAL PUBLICATIONS COMMENT FORM

We would appreciate your taking the time to fill out this form. Your answers and comments will help us improve the quality and usefulness of our publications.

Your Name ______________________ Your Title ______________

Company ____

Street ____

City ____________________________ State ________ Zip ________

Manual Title ______________________ **Manual No.** ______________

YES	NO	
☐	☐	Was the manual easy to use? If not, please explain why.

FOLD

☐	☐	Did the manual tell you everything you needed to know? What additional information would you have liked?
☐	☐	Was the information accurate? If not, please tell us what the inaccuracies were.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 26 SOUTHBORO, MA 01772

POSTAGE WILL BE PAID BY ADDRESSEE:

Data General Service, Inc.
A Subsidiary of Data General Corporation

F.E. TECHNICAL INFORMATION SERVICES
MS 125
50 Maple Street
Milford, MA 01757-9901

Data General Service, Inc.
A Subsidiary of Data General Corporation

014-000744-04